Enhanced Vision-Based Localization and Control for Navigation of Non-holonomic Omnidirectional Mobile Robots in GPS-Denied Environments

Mostafa Sharifi

Department of Mechanical Engineering

University of Canterbury

A thesis submitted for partial fulfillment of the degree of

Doctor of Philosophy in Mechanical Engineering

2017

This thesis is dedicated to

My Dear Parents

for their endless love, support and encouragement.

Acknowledgements

I would like to express my sincere appreciation to my Senior Supervisor, Professor XiaoQi Chen from Mechanical Engineering Department at University of Canterbury (UC) for providing the opportunity to study and mostly his guidance and support throughout my entire PhD study. Also my huge appreciation to my Co-Supervisory team from the UC department of Mechanical Engineering, Dr. Chris Pretty for his invaluable assistant and support throughout my research, and Dr. Don Clucas for his guiadance and assistance.

My gratitude to the technical staff of the UC Mechanical Engineering workshop, Gerry Kirk, Julian Murphy, David Read, Scott Amies, and Garry Cotton, who contributed essential technical work, guidance throughout this study.

The work presented in this thesis was financially supported through the departmental scholarship and fundings; both were generously awarded by the UC Department of Mechanical Engineering.

I would like to thank my dear guitars an my muso friends, Matt, James, Matthew, Mike, and Pete for the weekly fun and inspirational music time at the jam place. Furthuremore my music band mates including John, Peter, Robbie, and Vinni for their support and fun times with “The Sensational Old/Hot Dogs” band.

I would also like to thank my good friends at UC, Farzin, Reza, Samuel, Arun, Jarrod, and Natalia for the friendship, useful discussions and encouragement throughout these years.

Finally, I would like to express my special gratitude and appreciation to my family, my father and mother, my brothers ans sisters, for their endless love ans support. Last but not least,

my special sincere appreciation goes to my brother-in-law, Alireza, for his invaluable financial and intellectual support, whom without his support and inspirations, this achievement would not have happened.

Abstract

New Zealand's economy relies on primary production to a great extent, where use of the technological advances can have a significant impact on the productivity. Robotics and automation can play a key role in increasing productivity in primary sector, leading to a boost in national economy. This thesis investigates novel methodologies for design, control, and navigation of a mobile robotic platform, aimed for field service applications, specifically in agricultural environments such as orchards to automate the agricultural tasks.

The design process of this robotic platform as a non-holonomic omnidirectional mobile robot, includes an innovative integrated application of CAD, CAM, CAE, and RP for development and manufacturing of the platform. Robot Operating System (ROS) is employed for the optimum embedded software system design and development to enable control, sensing, and navigation of the platform.

3D modelling and simulation of the robotic system is performed through interfacing ROS and Gazebo simulator, aiming for off-line programming, optimal control system design, and system performance analysis. Gazebo simulator provides 3D simulation of the robotic system, sensors, and control interfaces. It also enables simulation of the world environment, allowing the simulated robot to operate in a modelled environment. The model based controller for kinematic control of the non-holonomic omnidirectional platform is tested and validated through experimental results obtained from the simulated and the physical robot.

The challenges of the kinematic model based controller including the mathematical and kinematic singularities are discussed and the solution to enable an optimal kinematic model

based controller is presented. The kinematic singularity associated with the non-holonomic omnidirectional robots is solved using a novel fuzzy logic based approach. The proposed approach is successfully validated and tested through the simulation and experimental results.

Development of a reliable localization system is aimed to enable navigation of the platform in GPS-denied environments such as orchards. For this aim, stereo visual odometry (SVO) is considered as the core of the non-GPS localization system. Challenges of SVO are introduced and the SVO accumulative drift is considered as the main challenge to overcome. SVO drift is identified in form of rotational and translational drift. Sensor fusion is employed to improve the SVO rotational drift through the integration of IMU and SVO.

A novel machine learning approach is proposed to improve the SVO translational drift using Neural-Fuzzy system and RBF neural network. The machine learning system is formulated as a drift estimator for each image frame, then correction is applied at that frame to avoid the accumulation of the drift over time. The experimental results and analyses are presented to validate the effectiveness of the methodology in improving the SVO accuracy.

An enhanced SVO is aimed through combination of sensor fusion and machine learning methods to improve the SVO rotational and translational drifts. Furthermore, to achieve a robust non-GPS localization system for the platform, sensor fusion of the wheel odometry and the enhanced SVO is performed to increase the accuracy of the overall system, as well as the robustness of the non-GPS localization system. The experimental results and analyses are conducted to support the methodology.

Publications

Journals

Sharifi M, Young MS, Chen X, Clucas D, Pretty C. Mechatronic design and development of a non-holonomic omnidirectional mobile robot for automation of primary production. Cogent Engineering, Taylor and Francis, 2016 Dec 31;3(1):125-431.

Sharifi M, Chen X, Clucas D, Pretty C., Cabon, E. Modelling and Simulation of a Non-Holonomic Omnidirectional Mobile Robot for Offline Programming and System Performance Analysis. Simulation Modelling Practice and Theory, Elsevier (In Revision).

Sharifi M, Schwärter I, Chen X, Pretty C. Fuzzy Singularity Avoidance for Optimal Kinematic Control of Non-Holonomoic Omnidirectional Mobile Robots, Robotics, MDPI publication(Submitted).

Sharifi M, Chen X, Pretty C. A Machine Learning Approach to Enhance Stereo Vision Based Odometry for Localization in Outdoor GPS-Denied Environments (Prepared).

Conference Proceedings

Sharifi M, Chen X, Pretty CG. Experimental study on using visual odometry for navigation in outdoor GPS-denied environments. In Mechatronic and Embedded Systems and Applications (MESA), 2016 12th IEEE/ASME International Conference on 2016 Aug 29 (pp. 1-5). IEEE.

Sharifi M, Young MS, Chen X, Clucas D. Mechatronic Design and Development of an Omnidirectional Mobile Robot for Automation of Primary Production. In Proceedings of International Conference on Innovative Design and Manufacturing ICIDM 2016.

Sharifi M, Chen X. A novel vision based row guidance approach for navigation of agricultural mobile robots in orchards. In Automation, Robotics and Applications (ICARA), 2015 6th International Conference on 2015 Feb 17 (pp. 251-255). IEEE.

Sharifi M, Chen X. Introducing a novel vision based obstacle avoidance technique for navigation of autonomous mobile robots. In Industrial Electronics and Applications (ICIEA), 2015 IEEE 10th Conference on 2015 Jun 15 (pp. 817-822). IEEE.

Note for Readers: Readers can access developed software for each part via personal contact.

Contents

List of Figures

List of Tables

Nomenclature

Roman Symbols

4WD4S Four Wheel Driving/Steering

AHRS Attitude and Heading Reference System

ANFIS Adaptive Neuro-Fuzzy Inference System

CAD Computer Aided Design

CAE Computer Aided Engineering

CAM Computer Aided Manufacturing

DOFs Degrees of Freedom

EKF Extended Kalman Filter

FIS Fuzzy Inference System

GNSS Global Navigation Satellite System

GPS Global Positioning System

GPU Graphic Processing Unit

ICP Iterative Closest Points

ICR Instantaneous Centre of Rotation

IMU Inertial Measurement Unit

KITTI Karlsruhe Institute of Technology and Toyota Technological Institute

libviso2 Library for Visual Odometry 2

LiDAR Light Detection and Ranging

MARIO Mobile Autonomous Rover for Intelligent

MBC Model Based Controller

MF Membership Function

ODE Open Dynamics Engine

RANSAC Random Sample Consensus

RBF Radial Basis Function

RMSE Root-Mean Square Error

ROS Robot Operating System

RP Rapid Prototyping

RTK Real Time Kinematic

SBA Sparse Bundle Adjustment

SDK Software Development Kit

SLAM Simultaneously Localization and Mapping

Sonar Sound Navigation And Ranging

SVO Stereo visual odometry

TSK Takagi-Sugeno-Kang

URDF Unified Robot Description Format

VO Visual Odometry

WMR Wheeled Mobile Robot

WO Wheel Odometry

Chapter 1

Introduction

1.1 Overview and Objectives

Application of mobile robots from indoor to outdoor usage in industries, agriculture and other sectors has been growing vastly. For such applications that are hazardous or hardly accessible, where it can be dangerous or difficult for humans, mobile robots can be the best choices to carry out the tasks. Primary production in New Zealand is one of the most important sectors. The success of New Zealand's economy which is mostly based on the success of primary production industries requires more productivity and efficiency [5]. Employing automation and robotics is one of the best solutions.

Common agricultural task that can be automated are including harvesting, weeding, spraying, and transportation. Many research has been carried out in the development of mobile robotic systems to automate different agricultural tasks for greenhouses [66][55], orchards [95][4], and agricultural fields [108][76]. The existing trend within primary production is to use bigger, heavier vehicles to perform agricultural operations in the shortest possible time. An alternative is to let smaller, light weight mobile robots cooperate and serve a team of workers. Such a robotic system achieves high productivity, lower cost and lower soil compaction which will make the production and operations more sustainable. Therefore, developing a robotic

system as a mobile base platform is required to augment and automate tasks in agriculture.

This thesis investigates design and development of a mobile robotic platform for application in unstructured environments such as orchards or agricultural fields. This platform is called MARIO - Mobile Autonomous Rover for Intelligent. The mechatronic design and development process of such a mobile robotic platform requires hardware and software development. Hardware including mechanical, electronics, and sensory systems to software including the control, sensing, and navigation system design.

The key challanges addressed in this thesis are:

- Optimal design, development, and control of the non-holonomic omnidirectional mobile robotic platform;
- Investigation, implementation, and improvement of non-GPS localization methods for navigation of the platform in GPS-denied environments;

1.2 Thesis Contribution

The main contributions of this thesis can be summarized into two groups follows:

Optimal design and control of the non-holonomic omnidirectional mobile robotic platform

1. *Design and development of the platform.* As the basic component of the thesis, mechatronic system design and development of a four wheel driving/steering (4WD4S) mobile robot as a non-holonomic omnidirectional platform is proposed. The design process is presented through an innovative integrated application of CAD/CAM/CAE and RP for rapid development of the robot. 3D CAD design allows further CAE analysis including motion analysis of the actuation system and structural analysis of the robot parts. CAM and RP are utilised to reduce the manufacturing time and cost. Software development

using Robot Operating System (ROS) in the design process enables design of the optimum embedded system to control the robot. The integrated design approach provides successful prototyping of the mobile robot.

2. *3D modelling and simulation*. The 3D modelling and simulation of the non-holonomic omnidirectional mobile robot, aiming for off-line programming and system performance analysis is investigated. For this purpose, platform is modelled and simulated based on the physical developed model using the Gazebo simulator and ROS. The singularity problem associated with the kinematic model-based control system is discussed. The singularities are introduced in form of representational (mathematics) and kinematic singularities. Mathematical singularities are solved by presenting the idea of switching between different steering scenarios. Simulation of the world environment, physical model, sensors, and control system are achieved in Gazebo simulator. The model-based kinematic controller is formulated, simulated, and validated with respect to the physical system. Modelling and simulation allows development, testing and validation of the robotic system and required software before implementation on the real system.

3. *Optimal model-based kinematic controller*. The kinematic singularity associated with the developed model-based controller is discussed. Kinematic singularities occurs in one form of steering, which places the Instantaneous Centre of Rotation (ICR) on one of the steering axes. This kinematic singularity damages the actuator and the steering axis, and should be avoided. To overcome that, a novel approach using fuzzy logic control is proposed to treat this type of singularity for the platform. The proposed methodology can be generalized for all non-holonomic omnidirectional mobile robots.

Enhanced localization for navigation of the platform in outdoor GPS-denied environments

1. *Vision based localization performance evaluation.* Experimental study and analysis on

the potential of using vision based localization for pose estimation of the platform in outdoor GPS-denied environments are carried out. Stereo visual odometry (SVO) is utilised and the SVO drift improvement as the main challenge in form of rotational and translational drift is investigated. Sensor integration is proposed as a solution to minimize the rotational drift.

2. *Enhanced vision based localization through machine learning*. The SVO translational drift and the nature of the SVO drift are studied. A Neural-Fuzzy machine learning model is proposed to model a translational drift estimator for the SVO. The model is designed based on the formulation of the parameters that represents the characteristics of SVO. The model is used to correct the translational drift at each frame and overally reduce the SVO drift.

3. *Sensor fusion for reliable localization in outdoor GPS-denied environments*. Experimental study and analysis of utilising sensor fusion to achieve more accurate localization of MARIO in GPS-denied environments. Kalman Filtering enhances the pose estimation of the platform by fusing the state information from wheel odometry (WO), Inertial Measurement Unit (IMU), and the corrected SVO.

The enhanced vision based localization for navigation in GPS-denied environments can be considered as the most important contribution of this thesis. However, design and development of the platform as the base tool of this research is essential to carry out the rest of the research work. Design and development of the robotic platform is done by considering the main challenges and improving those in terms of the design and control system. The developed method in enhanced localization can be applied to any other ground robotic platform by ignoring other criteria such as the specific design, locomotion, or control system used in those robotic systems.

1.3 Thesis Outline

This thesis is organised into nine chapters. The first chapter as presented here introduces the motivation, research objectives, and the contribution of this research. The next chapters are as follows:

Chapter 2. *Literature Review* presents the overview of the recent research and challenges in developing the mobile robotic platforms for operation in unstructured outdoor environments, specifically agricultural environments. The review involves research carried out in the design, control, and navigation systems with the specific focus on localization.

Chapter 3. *Development of Non-holonomic Omnidirectional Mobile Robot* presents the mechatronic design and development process of the developed mobile robotic platform MARIO - Mobile Autonomous Rover for Intelligent Operations, a four-wheel driving/steering (4WD4S) as non-holonomic omnidirectional mobile platform.

Chapter 4. *Modelling and Simulation of Non-holonomic Omnidirectional Robot* presents the 3 dimensional (3D) modelling and simulation of MARIO, aiming for off-line programming and system performance analysis. The modelling and simulation approach allows successful development and test of the platform and different implemented robotic software in a simulation environment before implementation on the real system.

Chapter 5. *Fuzzy Singularity Avoidance for Optimal Kinematic Control* discusses the singularity problems of the 4WD4S platforms and provides a fuzzy control system to solve the problem of kinematic singularity for an optimal kinematic control system.

Chapter 6. *Enhanced Stereo Visual Odometry Using Sensor Fusion for Non-GPS Localization* presents the experimental study and analysis of utilizing stereo visual odometry for MARIO as a solution for localization and navigation in outdoor unstructured GPS-denied environment such as orchards as a main part of the research work in this thesis.

Sensor fusion using Extended Kalman Filter is utilised to study the improvement of the localization of the system by the fusion of stereo visual odometry and the IMU. The challenges and drawbacks of vision based localization in terms of accuracy and drift are discussed and analysed.

Chapter 7. *Enhanced Stereo Visual Odometry Using Machine Learning* presents a novel machine learning approach to enhance the stereo visual odometry by reducing the drift with the presented experimental results. This includes a more detail explanation of the stereo visual odometry and analysis of the nature of the drift in the vision based localization systems. The Neural-Fuzzy based machine learning system is formulated as a drift estimation system to then correct the odometry drift and improve the accuracy of the stereo visual odometry system.

Chapter 8. *Enhanced Localization for Navigation in GPS-Denied Environments* presents the integration of the developed methods including the sensor fusion of the wheel odometry, IMU, and the enhanced visual odometry to achieve a more accurate and reliable localization for navigation in GPS-denied environment.

Chapter 9. *Conclusion and Future Works* presents the summary of the work presented in this thesis alongside the recommendation and addressed challenges for future research.

Chapter 2

Literature Review

This thesis investigates design and development of a mobile robotic platform for potential use in agricultural applications. This Chapter discusses the background research and motivation, in particular, the challenges relating to the agricultural mobile robotics and in general, field service robotics in terms of locomotion, modelling, control, and navigation.

2.1 Agricultural Mobile Robotics

The review of the trend in agricultural mobile robots shows an ascending progress during the last few years. An autonomous Kiwi-Fruit picking robot was developed at Massey University [92] for automation of kiwi fruit production in New Zealand. The vehicle has four wheels and uses two front wheels for steering. This vehicle uses differential Global Positioning System (GPS), compass and computer vision for localization and navigation in kiwifruit orchards. U-Go robot was developed by the Service Robots Group of University of Catania [7] as a multi-functional system to cover different field of applications especially precision farming applications. The locomotion is provided by the rubber tracks instead of wheels and each track is driven by a DC motor to provide differential drive skid steering. Localization is provided by Global Navigation Satellite System (GNSS) receiver and X-Sens MTi Attitude

and Heading Reference System (AHRS). Data from stereo camera, Sick LMS200 Laser Range Finder and Sound Navigation And Ranging (Sonar) sensors are fused to provide information for navigation algorithm.

(a) (b)

Figure 2.1: a) Kiwi-Fruit picking robot [92], b) U-Go Robot [7].

Zeigbee robot developed by Bio-Mechatronics group at National Pingtung University of Science and Technology, Taiwan [18] is a semi-autonomous mobile robot gardener prototype for spraying in the greenhouse. This robot was designed with three wheels, driven and steered by the front-wheel. Localization was performed by wheel odometry while navigation including row detection, row following and obstacle avoidance was provided by laser range finder and ultrasonic sensors. VineRobot (Figure 2.2a) project is the result of collaboration between different universities in Europe which has received funding from the European Union's Seventh Program [27], is a mobile robotic platform which is equipped with several non-invasive sensing technologies to inspect and monitor agronomical and physiological parameters of the vineyards. Bin-Dog robot (Figure 2.2b) is an intelligent fruit bin carrier developed at Carnegie Mellon University, USA by [111] to travel autonomously in orchards and move/place fruit bins for fruit pickers to allow for efficient harvest. A four-wheel independent steering (4WIS)

system provides steering modes such as four wheel coordinated steering, crab steering and spinning. Localization and navigation outside the rows are provided using differential GPS and inside rows using laser range finder and ultrasonic sensors.

(a) (b)

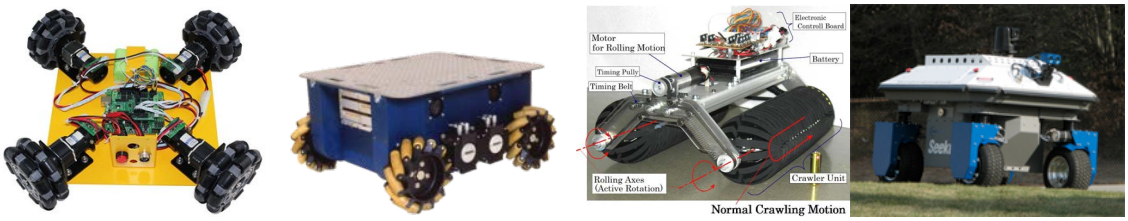
Figure 2.2: a) VineRobot [27], b) Bin-Dog Robot [111].

2.2 Locomotion of Ground Mobile Robots

Ground mobile robots can be categorized based on the locomotion system into legged, wheeled, tracked and hybrid robots [12]. These require different mechanical and control system designs. Wheeled and tracked mobile robots are the most used in field service applications. Generally, the locomotion control system of mobile robots can be divided into differential drive, car-like, and omnidirectional systems [105]. The existing off-road locomotion systems for mobile robots are mostly based on the conventional tracks or wheels with car like or differential drive steering systems. With the current off-road locomotion systems, navigation and mobility in narrow and restricted spaces such as orchards and farm rows are difficult. An omnidirectional mobile robot fulfils these abilities very well. This type of robot is capable of moving in any dir-

ection while keeping the platform orientation constant, which results in significant advantages over the conventional platforms in terms of agile mobility.

There are different omnidirectional robot designs (Figure 2.3) which use different types of wheels such as omni-wheels [6], mecanum wheels [49], conventional wheels in form of independent steering wheels [71], and omnidirectional tracks [99]. Proper design requires a good understanding of the application and environments that a robot should function. Omni-wheeled or mecanum wheeled systems are limited to use for off-road applications by their design and passive rollers and they need a completely flat surface. The omnidirectional tracked crawler requires a complex mechanical design and is limited to provide a uniform omnidirectional motion.

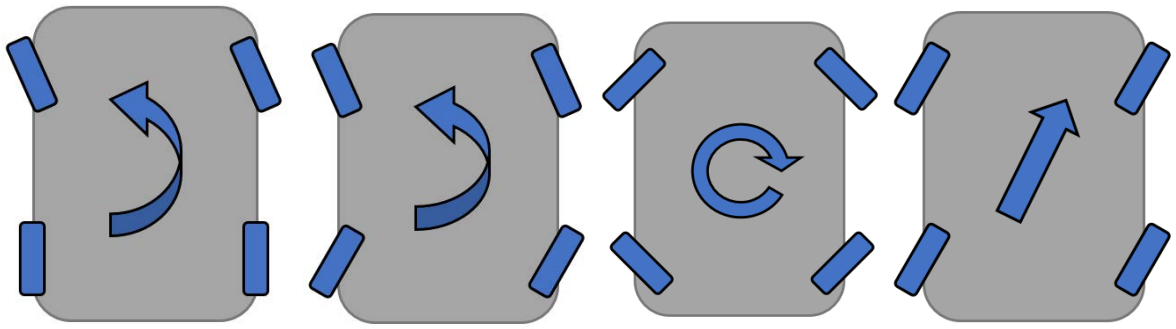


(a) (b) (c) (d)

Figure 2.3: omni-wheels robot (obtained from nodna.de); (b) URANUS mecanum wheels omni-directional robot (obtained from wikipedia.org/wiki/Mecanum_wheel); (c) Omni-Crawler (obtained from hh.mech.eng.osaka-u.ac.jp); (d) Seekur 4WD4S (obtained from mobilerobots.com).

Independently steered wheels are the only solution for off-road omnidirectional mobility. Wheeled mobile robots are the best known types of mobile robots for field service applications. Example of wheeled mobile robots for field service applications are Seekur, an omni-directional all-terrain mobile robot developed by Adept Mobile Robotics [70]; Hortibot, an agricultural robot developed in Institute of Agricultural Engineering at University of Aarhus [85]; Bonirob, an autonomous field robot developed by AMAZONE and Osnabrück University [88]. A four-wheel independent steering and driving (4WD4S) robot design allows for

the setup of multiple steering configurations including car-like steering, coordinated steering, spot turn, and omnidirectional (crab) steering system (Figure 2.4).



(a) (b) (c) (d)

Figure 2.4: 4WD4S steering modes: (a) car-like steer; (b) coordinated steer; (c) spot turn steer; (d) omnidirectional/crab steer.

2.3 Modelling, Simulation, and Control System

Nowadays modelling and simulation are the major parts of scientific and engineering processes, especially robotic systems. Modelling and simulation play an important role in off-line programming, performance analysis, and development of advanced control algorithms for robotic systems [103]. Design, testing, and validation of robotic systems ranging from indoor [65] to outdoor mobile robots [63], articulated industrial manipulators [107], underwater robotic systems [90], and humanoid robots [61] are impossible without proper modelling and simulation tools. Furthermore, simulation can be used as a tool to develop virtual environments for training operators [84] as well as an educational tool for teaching and learning basic concepts of robotic systems [14]. Additionally, simulation provides low cost means of testing and experimentation, and makes controlling disturbances much easier compared with using real robotic systems [102].

2.3.1 Robotics Simulation and Software Development Tools

Along with development and progress in powerful and affordable computing technologies in last two decades, a number of proprietary and open source robotic modelling and simulation software have been developed. Examples of open source robotic simulator software which have achieved popularity among users and are available freely for personal or academic use are Open Dynamics Engine (ODE) [30], Robotic Toolbox for MATLAB [26], Microsoft Robotics Developer Studio (MRDS) [50], Webots [69], Virtual Robot Experimentation Platform V-Rep [86], Modular Open Robots Simulation Engine MORSE [36], and Gazebo [56]. Selecting the most suitable simulation tool for a specific purpose like research, development, or education can be difficult. The variety of simulation tools, features provided by each tool, user-friendliness and dependency on external packages are some of the main considerations which can make it challenging to choose the most suitable robotic simulation tool [103].

The Gazebo Project was a part of the Player/Stage/Gazebo projects, which have been developed since 2001. The Player Project provides a network interface to different types of robots and sensors. Stage is a simulator for a large population of mobile robots in a 2D environment [68]. Gazebo is a multi-robot simulation tool which has the capability of accurate and efficient simulation of a population of robots, sensors and objects in a 3 dimensional world. Gazebo generates realistic sensor feedback and has a robust physics engine to generate interactions between objects, robots and environment. Furthermore, Gazebo provides high-quality graphics, and suitable programmatic and graphical user interfaces [56]. Gazebo is offered freely as a stand-alone software, but has also been packaged along with Robot Operating System (ROS) as the simulation tool. ROS was originally developed by the Stanford Artificial intelligence Laboratory in Support of the Stanford AI Robot (STAIR) project. ROS is an open source robotic middle-ware that provides libraries and tools to help software developers produce robot programs. It provides hardware abstraction, device drivers, libraries, visualizers, message-passing, package management, and more [82].

There have been many robotics research projects where Gazebo/ROS has been used widely. BoniRob is an autonomous field robot platform for individual plant phenotyping and Gazebo has been used to model and simulate the farm environment and robotic model [88]. Linner and Shrikathiresan presented modelling and operating robotic environments using Gazebo/ROS with some common examples to prove the performance and effectiveness of using ROS and Gazebo for this aim [59]. A comprehensive simulation of quadrotor UAVs has been carried out using ROS and Gazebo [67] and is able to simultaneously simulate different features such as flight dynamics, on-board sensors like IMUs, external imaging sensors and complex environments. A set of simulations for manipulation tasks using ROS and Gazebo have been completed to illustrate the techniques of implementing robot control in a short time [81].

2.3.2 Non-holonomic Omnidirectional Control System

The degree of holonomy of a Wheeled Mobile Robot (WMR) is defined by the degrees of mobility and manoeuvrability. WMRs are classified into holonomic and non-holonomic classes based on their degrees of mobility and steer-ability [62]. A holonomic WMR is capable of moving in all available degrees of freedom (DOFs) in a workspace and it is not limited by any mechanical constraint to move. A non-holonomic WMR is constrained by its mechanical structure and kinematic presentation, thus cannot move in all directions without changing the steering configuration and discontinuity in the motion.

The kinematic modelling of 4WD4S as a non-holonomic omnidirectional platforms is achieved using the concept of instantaneous centre of rotation (ICR). These platforms are specified by having a degree of steer-ability of two, and a degree of mobility of one[15]. The ICR is defined as the rotation point for the platform with respect to the centre of robot frame. The kinematic modelling of the non-holonomic omnidirectional WMRs based on the ICR can be achieved using the Cartesian representation of the ICR [28], or the polar representation of ICR [24]. Each of these representations has its own mathematical and kinematic singularities.

Mathematical singularities are resulted by having null angular or linear velocities, causing singularity for ICR vector elements. This type of singularity can be solved by introducing different type of steering modes for the platform and switching between these modes.

Kinematic singularity results when the ICR passes through any of the steering joint axes. In this situation, the steering angle of the wheel for the respective joint cannot be individually defined as infinite number of solutions for steering angle are exist. Also due to kinematic singularity, the steering rate of the joint in singular region increases unboundedly, gets over the mechanical and electrical limits of the actuator and can damage it. To deal with kinematic singularity, different methods have been proposed. A group of methods try to keep or push the ICR out of the singular regions. Artificial Potential field methods have been used in [25, 28, 94] to push the ICR from placing or moving nearby the steering axes. The other group of methods presented in [23, 77] consider limiting velocity or acceleration for the situation that the ICR is passing on a path over the joint steering axes or stopping at the steering axes, so that more velocity workspace is available.

2.4 Vision Based Localization and Sensor Fusion

Navigation is one of the most challenging areas in mobile robotics research. Success in mobile robot navigation obliges success in four building blocks of navigation system including perception, localization, cognition and motion control. Localization, defined as the ability of a robot to determine its position in a 2D or 3D environment, is one of the main building blocks of the navigation system of any mobile robotic system and it has received the most research attention as the basic step towards reliable navigation [96].

2.4.1 Localization

GPS is believed to be the most common method to achieve localization in outdoor environment, has been used for many years as the main core of the navigation system in different applications. GPS has been used widely from the unmanned ground vehicles [16, 110]to unmanned aerial vehicles [87, 106]. While GPS seems to be the most promising solution for localization in outdoor environment, it suffers from signal occlusion which results precision reduction or lost positioning information. This loss can be caused by the buildings in an urban environment, tree canopies in an orchard environment, or any other object blocking the signals between the satellites and the GPS receiver antenna. Moreover, without signal blockage, High precision GPS units are excessively expensive, thus not affordable for many applications. Furthermore, GPS alone does not provide inertial information, so an IMU is always required to provide inertial and rotational information [33].

Other localization approaches are considered as alternatives or compliments to GPS. Utilising wheel odometry and IMU to localize robots add no extra cost as they are required for the motion control system. However, both are prone to drift and error accumulation over time, so it leaves them currently unreliable to be used for localization [20]. Recently, vision based techniques have become the subject of research for localization.

Light Detection and Ranging (LiDAR) sensors have become one of the most attractive solutions for localization due to their relative low cost (compare to high accuracy GPS) and high accuracy [45]. They can be used to identify and localize landmarks based on a provided map [10] or simultaneously building a map and localizing (SLAM) the robot in that map . Lasers not only can be employed for mapping and localization but also for several other purposes such as row and obstacle detection/extraction for navigation system [9]. Scan matching is performed between two sets of scans and relative displacement and orientation can be extracted. Traditionally pose estimation between two lasers scans could be computed using Iterative Closest Points (ICP) algorithm [80]. However due to large number of iterations, ICP is

computationally expensive and also it diverges from global minimum without having a proper initial estimation of the transformation matrix. ICP is still the core algorithm for scan matching with different representations to optimize the performance of ICP. Point-to-point [60] and point-to-plane [79] are the most common ICP based techniques. Their performance are very good for structured environment but the research challenges still remain for unstructured environment [80].

2.4.2 Sensor Fusion

To accurately estimate the robot's position, data fusion and integration of information provided by different sensors is required. To deal with uncertainties caused by the measurement errors in sensors, probabilistic approaches have been used to provide a more accurate pose estimation of the robot. One of the common technique is Extended Kalman Filter (EKF) [101]. EKF has been used widely for localization of outdoor mobile robots to fuse Odometry and GPS data [100], Odometry, GPS and IMU [72] or integrating Odometry and laser data for SLAM [46].

Kalman Filtering is a common technique used for estimation of accurate observed sensor data including noise and inaccuracy over time. The Kalman Filtering algorithm works recursively in two steps: prediction and update. In the prediction step, an estimation of the current state and covariance is produced based on the previous state estimate. In the update step, the previous estimate is combined with the current observation to produce an estimation of the current state and covariance. The EKF is the non-linear version of the Kalman Filter, where the state estimation and observation models are non-linear functions[101].

2.4.3 Visual Odometry for Localization

Vision based navigation systems have been the subject of research interest in the last two decades and many vision based techniques have been developed and tested. The application of these techniques are utilised based on their ability to process and extract the visual features

from the sequential digital images and estimating the displacement and rotation based on those features [8]. The term visual odometry (VO) was first introduced in [75] as the process of ego-motion estimation of a robot using cameras. The motion is estimated based on the triangulated extracted feature machetes of the consecutive frames from the image sequences (Figure 2.5).

VO has been widely developed and tested using monocular cameras [1, 21], stereo cameras [22, 93], and omnidirectional cameras [57, 91]. One of the well-known successful application of VO for localization in a GPS-denied environment has been the series of Mars Exploration Rover missions [19]. VO also have been broadly used from unmanned ground vehicles [42, 97], to unmanned aerial vehicles [13, 17], and unmanned underwater vehicles [2, 35]. Despite the fact that VO gives a reasonable estimate of the motion, one of the main problems of VO in long range navigation is drift which is caused by the growth of small errors in the motion estimation process at frame over time. The source of the VO drift is mainly the uncertainties in the feature matching process [47].

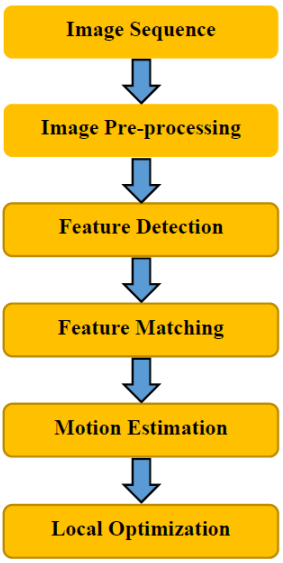


Figure 2.5: VO processing steps.

To overcome this problem, different solutions have been proposed. These solutions can be

categorised in two classes; the ones using sensor fusion, and the non-sensor fusion computational techniques. The sensor fusion based methods (as mentioned earlier) use probabilistic approaches to deal with the measurement uncertainties and errors from different sensors and provide a more accurate motion estimation. Common techniques such as Extended/Unscented Kalman filters, [3] or Particle filters, [109] have been implemented to integrate motion information from different sensors such as VO, IMU, wheel odometry, and GPS data.

Non-sensor fusion drift minimization methods focus on reducing the drift to increase the accuracy of VO system by implementing computational or probabilistic methods on the VO system. Different techniques have been proposed to reduce the drift. Bundle adjustment [104] is a common method that has been used widely to minimize the VO drift. It reduces the error accumulation by simultaneously optimizing the camera parameters, the 3D coordinates presenting the scene geometry, and the relative estimated motion. In this case, image features are tracked over more than two frames, and the optimization is performed by minimizing the image reprojection error. The reprojection error is a non-linear function and real time long range full bundle adjustment is almost impossible due to the huge number of features and 3D estimated poses. To overcome this issue in bundle adjustment, sparse bundle adjustment (SBA) has been proposed [98] which reduces the computational complexity of the optimization process by not considering any uncertainty of whatever kind.

Sakai et al. in [89] propose a technique based on Hybrid Neural Fuzzy Inference (HyFIS) system to learn the noise pattern of VO system. To train this HyFIS, they consider 3 parameters of the feature points as inputs to the learning system. These parameters are including the inlier numbers, average distances between 3 feature points, and variance of interior angles between those 3 feature points. Those 3 feature points are the randomly selected samples as used for 3-point algorithm in motion estimation. The output of the HyFIS is the error between the VO estimation and GPS as the ground truth. With this approach, they could improve the VO estimation for two runs of experiment, $45\,m$ at first run and $70\,m$ at the second run. The error

compensation has not been quantified.

More recently, drift was seen and introduced as a bias in the estimates that grows over time. Dubbelman and Groen in [31] investigated the existence of a bias in motion estimation process of a stereo VO system. They identified the distribution and incorrect modelling of uncertainties of the 3D coordinates as the main sources of bias, causing the estimator to drift over time. In their future work (Dubbelman et al. in [32]), they developed a projective bias model to use for error compensation. What they suggest is to assume that the camera model used in their research is an approximation and they consider this as the source of the bias. They use trajectory calibration to estimate parameters of a bias model off-line and use it to compensate for bias in the online process. Rehder et al. in [83] introduce another technique to estimate the VO bias which is based on the Monte Carlo simulation which is computationally expensive. This has been performed by simulating a Gaussian noise model on the 2D image feature locations for a huge number of samples. Bias is estimated as the norm of the projected translation vector with the added Gaussian noise over the original estimated solution.

Farboud-Sheshdeh et al. in [38] dig more into the nature of the bias in VO and propose an online method based on the modified sigma-point method for sampling and compare the results to the typical Monte Carlo sampling and an existing analytical method. Based on their simulations, bias estimation is achieved with the same accuracy as the Monte Carlo, but at a fraction of the computational cost. Finally they employ the idea of bootstrapping in statistics to estimate, correct, and reduce the bias online.

2.5 Summary

This chapter has reviewed literature from a wide varaiety of key topics related in design and development of a mobile robotic platform, capable of operation in field service applications with focus on agricultural environments such as orchards. The detailed review on the locomotion system and 4WD4S type as the optimal locomotion, introduced challanges in design

and control for this type of locomotion system. The associated singularity problems for control system design of the 4WD4S platform as a non-holonomic omnidirectional system were described. The importance of a reliable localization system as the core part of the navigation system for operation of the robotic platform in GPS-denied environemnts such as orchards was reviewed. Vision based localization and its challanges as a choice for non-GPS localization were discussed. These challanges and the state of the art related to each of them as well as the research gap were addressed. Each of these challanges and the solutions are presented in the next chapters.

Chapter 3

Development of Non-Holonomic Omnidirectional Mobile Robot

3.1 Introduction

This chapter presents the mechatronic system design and development of the four-wheel drive/steer (4WD4S) mobile robot as a non-holonomic omnidirectional robot, MARIO - Mobile Autonomous Robot for Intelligent Operations. An innovative integrated application of Computer Aided Design (CAD), Computer Aided Manufacturing (CAM), Computer Aided Engineering (CAE), and Rapid Prototyping (RP) has been employed for rapid development of the robot chassis and other mechanical parts by using different software tools. Most of the parts were designed by 3D CAD software which allows further CAE analysis including structural and motion analysis. To reduce the manufacturing time and cost, CAM and RP have been used to manufacture the main parts. These master parts are manufactured by workshop machining, 3D printing, and laser cutting. Robot Operating System (ROS) has been utilized as the core robotic software development environment.

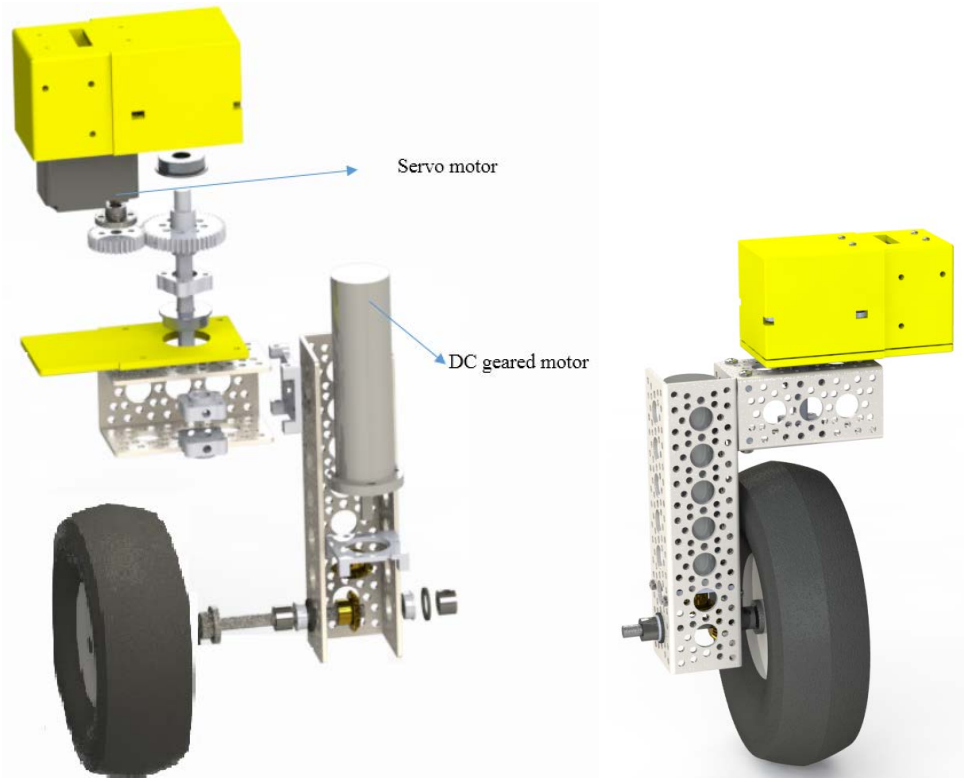
3.2 Mechanical Design and Development

3.2.1 Design Overview

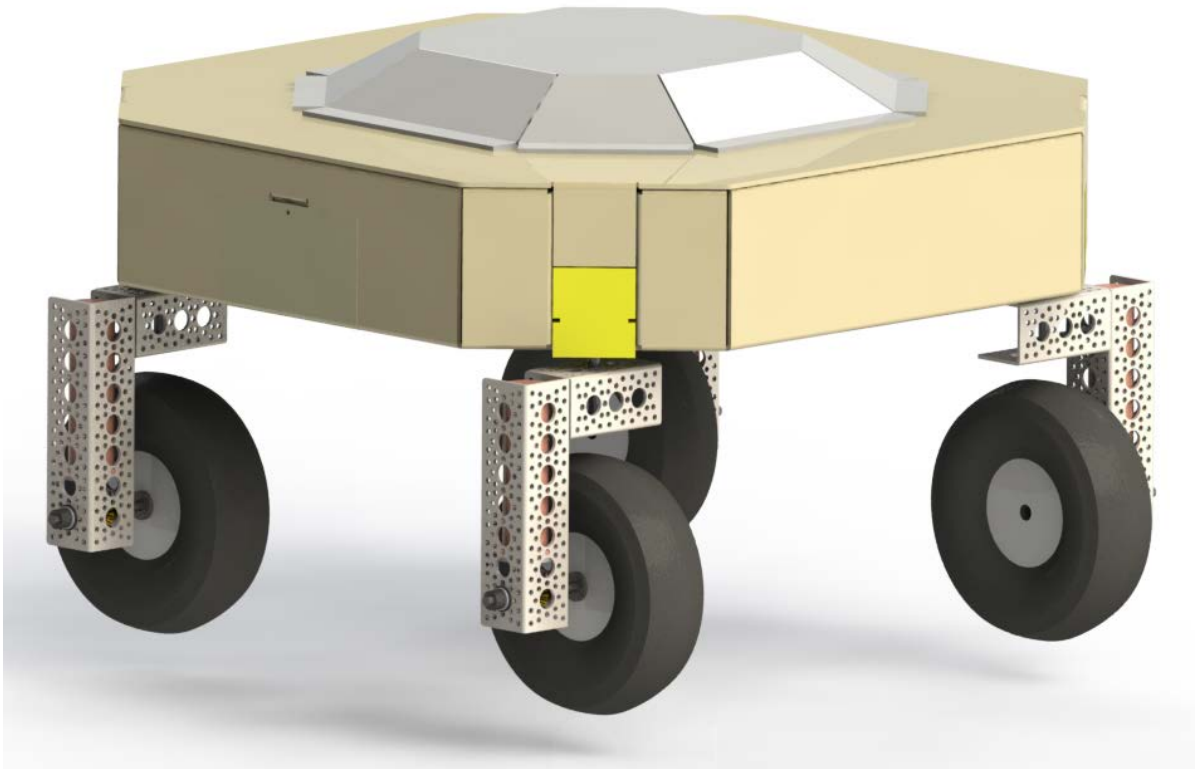
For the mechanical design of the robot, several design criteria need to be considered such as body weight and size, manufacturability, manufacturing time and cost. In the initial design process of this robot, it was aimed to have a modular system for easy assembly and further maintenance. As a four-wheel independent steering/driving mobile robot, development of the wheel module was the most important part of the design. This module includes the steering and driving mechanisms which is presented in Figure 3.1a.

The steering mechanism consists of a servo motor which provides steering torque, a set of gears, bearings and output shaft. All of these parts are placed in a housing. The driving mechanism consists of a DC motor, leg structure, a set of bevel gears, bearings and output shaft which drives the wheel. For this robot, four of these wheel modules attach to the chassis (built from aluminium box section) to form a 4WD4S mobile robot. This design approach allows initial analysis, tests and validation on a single wheel module and also decreases the production time and cost for all four wheel modules.

All the robot parts have been designed using SolidWorks (v. 2014, SolidWorks Corp.) 2014 CAD software to allow further CAM/CAE. This design includes the parts which should be manufactured and also the other parts supplied from off the shelf. Figure 3.1a shows the exploded view of a wheel module. The assembled wheel module is shown in Figure 3.1b. Figure 3.1c shows the final assembly of MARIO.



(a) (b)



(c)

Figure 3.1: 3D CAD model of MARIO; (a) exploded view of a wheel module, (b) assembled view of a wheel module, (c) Assembled MARIO.

3.2.2 CAE Analysis for Design Verification

3.2.2.1 Structural Analysis of Wheel Module

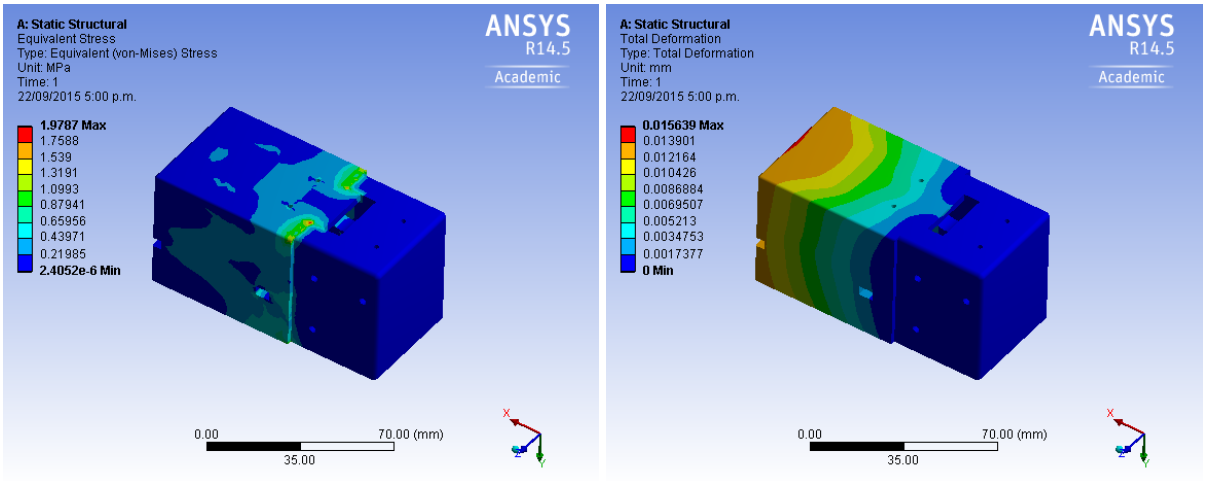
The servo housing part which includes all the steering mechanism parts is designed to be 3D printed using engineering plastics. The engineering plastic material Ingeo Polylactic Acid (PLA) was used. Table 3.1shows the mechanical properties of PLA. This part is connected to the chassis and also holds the driving mechanism to provide steering. Thus, there would be mechanical stresses on this part caused by the applied loads and forces. To check and validate the structural strength and integrity of the 3D printed housing part, CAE analysis has been conducted using ANSYS [v 14.5, ANSYS, Inc]. The finite element model of housing part is composed of 73304 nodes and 41965 elements. To run CAE analysis, it was assumed that the maximum applied load on this part is concentrated on the inside surface of servo housing part which holds the output shaft and the flanged bearing. The Applied force is considered $100N$ which is a heavy force compared to the total weight of the robot which is about 10 kg.

Table 3.1: Mechanical properties of Ingeo PLA [73].

Property	Content
Tensile yield strength (MPa)	60
Tensile ultimate strength (MPa)	53
Tensile modulus (GPa)	3.6
Poisson's ratio	0.39
Density(g/cm^3)	1.25

Figure 3.2 shows the structural analysis for the housing part including the distribution of von Mises stress and the total deformation. The stress is concentrated on the top side edges of housing part which are connected to the chassis. With the initial analysis, the maximum stress was 3.56 MPa. To reduce this stress, the design was modified by increasing the thickness of

those edges. The resulting maximum stress was reduced to 1.9787 *MPa* which is shown in Figure 3.2a. This modification decreased the maximum stress to 55 % of the maximum stress achieved by initial design and test. Figure 3.2b shows the total deformation analysis. The maximum deformation is 0.015 *mm* which is completely negligible. This type of analysis has been carried out for other parts to improve the structural stiffness and safety of each part and the assembled robot.



(a) (b)

Figure 3.2: CAE structural analysis for housing part; (a) Distribution of von Mises stress, (b) total deformation.

3.2.2.2 Motion Analysis for Joint Torque Validation

Robot motion is provided by 8 active joints which are driven by 4 servo motors for steering joints and 4 DC motors for driving joints. Required actuation Torque was calculated based on the parameters depicted in Table 3.2, and Figure 3.3 shows a simple free body diagram of MARIO and various associated forces that act on the platform. The required torque was calculated as 3.63 *N.m* for the whole platform based on the specifications provided in Figure 3.3 and Table 3.2 is defined as below and the formulation in [74]:

Table 3.2: Considered parameters in required total torque estimation.

Parameter	Value
Mass	$10kg$
Payload	$10kg$
Max Speed	$1m/s$
Max incline	10°
Max acceleration	$0.5m/s^2$
Driving wheel radius	$0.0825m$
DC motor speed	146 RPM

$$T = F_w * r, \tag{3.1}$$

$$\Sigma Forces = F_{total} = F_w - F_g = Ma, \tag{3.2}$$

So that:

$$T = M(a + g\,sin\theta)r \tag{3.3}$$

Where in Equations 3.1, 3.2, and 3.3T is the required torque, F_w is the force pushing against the wheel, $F_g$is the force pulling the robot down on the incline surface due to gravity defined with the acceleration of g, M is the platform mass and a is the platform acceleration. The friction force is ignored, as it is considered as the force required for the wheels to drive the robot forward. The required driving torque for each wheel is a quarter of the total torque as 0.9 $N.m$.

The maximum speed of each wheel and in total, the platform based on the DC motor

maximum RPM is 1.263 $^{m}/_{s}$ through below formulation:

$$V_{max} = \frac{2\pi r * RPM}{60}, \tag{3.4}$$

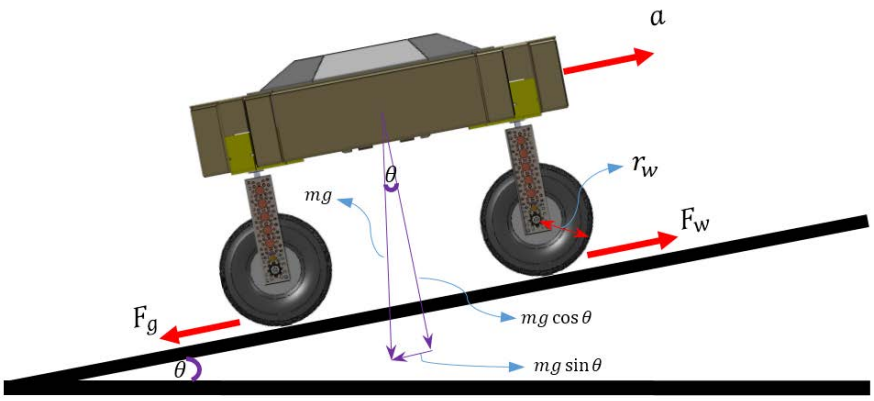
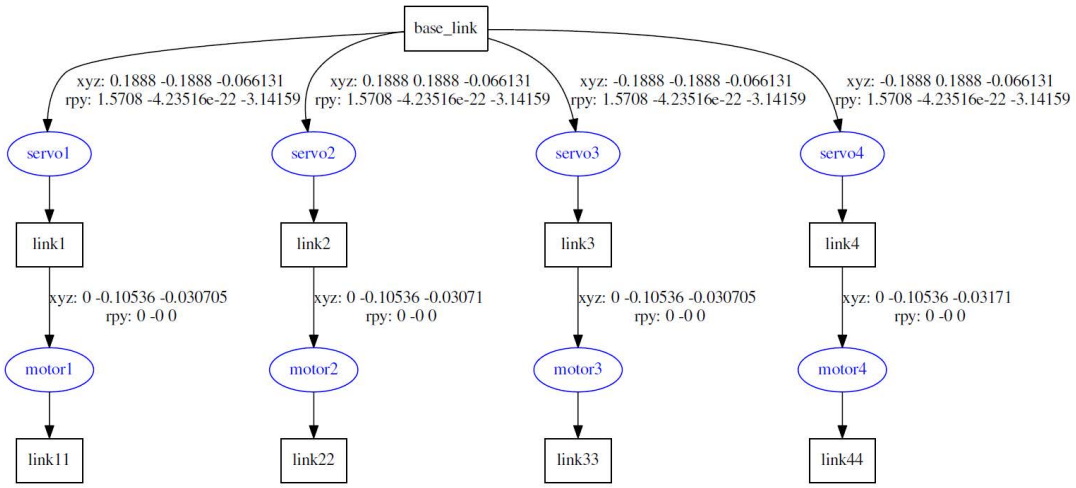


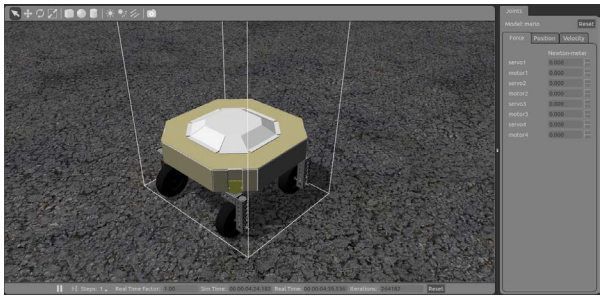
Figure 3.3: Simple free body diagram of the MARIO and the associated forces in the drive mode.

To validate these calculations, a motion analysis has been carried out using Gazebo Simulator . Gazebo is a robotic simulation tool which has the capability of accurate and efficient dynamical simulation of robots, sensors and objects in a 3D world. It generates realistic sensor feedback and has a robust physics engine to generate interactions between objects, robots and environment [18]. The 3D CAD design by SolidWorks can be imported into Gazebo. This can be achieved by exporting 3D model as a URDF file and the associated STL files. Unified Robot Description Format (URDF) is an XML file format to describe all kinematic and dynamic elements of a robot. The kinematic chain of the robot generated by URDF file is shown in Figure 3.4a. This robot has 9 links and 8 joints. In URDF terminology multiple links can be connected to one link by joints and defined parent and child links. Steering joints are defined as revolute joint while the driving joints are defined as continuous. This robot definition allows to use Gazebo for dynamic simulation. This robot has 8 DOFs in total on robot local frame which provides 3 DOFs motion (V_x ,V_y, and ω) in the world frame. Figure 3.4b shows

the simulated robotic model in the Gazebo environment.



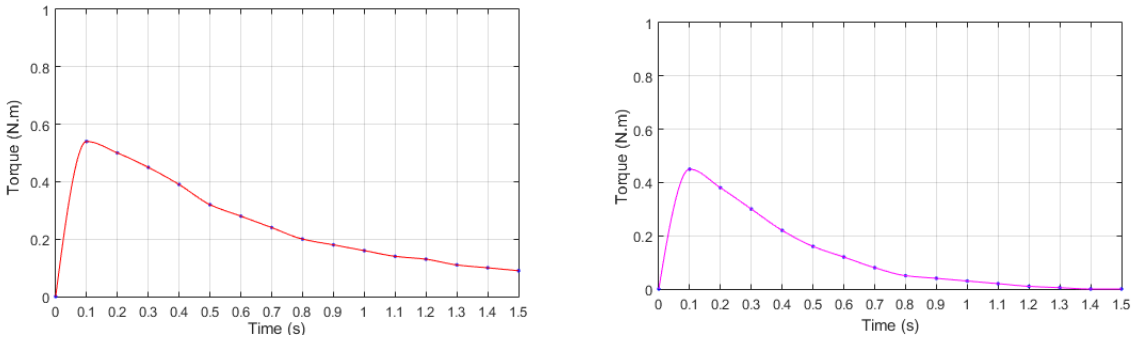
(a)



(b)

Figure 3.4: Motion analysis using Gazebo Simulator; (a) Kinematic chain of MARIO, (b) Gazebo environment and the simulated robot.

As an example, a forward motion and steering are simulated to analyse the response of the driving and steering joints. For the forward motion, all four wheels are driven with the same speed. The test speed was 1 m/s and the steering angle was 90 degrees for the duration of 1.5 s. These tests have been done separately. Figure 3.5 shows the resulting torque values for these analysis for one servo (Figure 3.5a) and one DC motor (Figure 3.5b). The rated torque for both DC motors and servos are 1 $N.m$. The acquired torque results are in the range of nominal torques for the servos and DC motors.



(a) (b)

Figure 3.5: CAE motion analysis for joint torque validation; (a) Driving joint torque response, (b) Steering joint torque response.

3.2.3 3D Printing for Rapid Prototyping

The design process of MARIO was focused on developing a robot by considering modularity, manufacturability and assemblability. This will provide easy maintenance and design modification for future design development. Some of the parts including the servo housing were made using 3D printing. 3D printing provides an easy solution for early stage prototyping to develop the required parts for the system by decreasing the manufacturing time, cost and inventory expenses. PLA material was chosen for 3D CAM of the servo housing part using 3D printer. RP allows rapid fabrication of these parts with very low cost of production (about 10 cent per gram). Figure 3.6 shows the 3D printed parts for servo housing. Laser cutting was also used for making the robot's body based on the 3D CAD. The initial model was laser cut out of medium-density fibreboard (MDF) material with the thickness of 5 mm, but perspex material was considered for the final prototype.

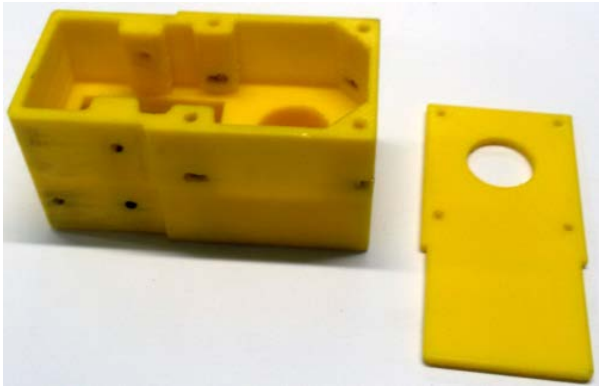


Figure 3.6: 3D printed servo housing parts.

3.3 Electronics and Software Architecture

3.3.1 Electronics and Sensory system

Each of the 4 wheel modules is equipped with a HerkuleX DRS-0101 Smart Robot Servo for steering and a 12V Low noise 146RPM DC Motor model 28PA51G. Both servos ans DC motors are equipped with encoders to measure the the rotation angle or speed.

The control system for the whole platform is provided using the master computer, Intel NUC which runs Robot Operating System (ROS) (v Indigo, Willow Garage) on Ubuntu 14.04 LTS. Figure 3.7 shows the component diagram of MARIO including the computers, sensors, communication and interfaces, and driving component. Servos and DC motor speed controllers are interfaced to the NUC through USB/RS232 communication for both driving commands and the encoder data. Sensory data for control and navigation is provided by wheel odometry from the wheel and servo encoders, a 9 DOFs Razor IMU, ZED (v 1.0, Stereolabs) stereo camera, and Swiftnav Piksi Real Time Kinematic (RTK) GPS unit which provides centimetre-level accuracy.

All sensor units are interfaced through USB communication to the NUC. ZED requires the Software Development Kit (SDK) and drivers run on a computer with a Graphic Processing

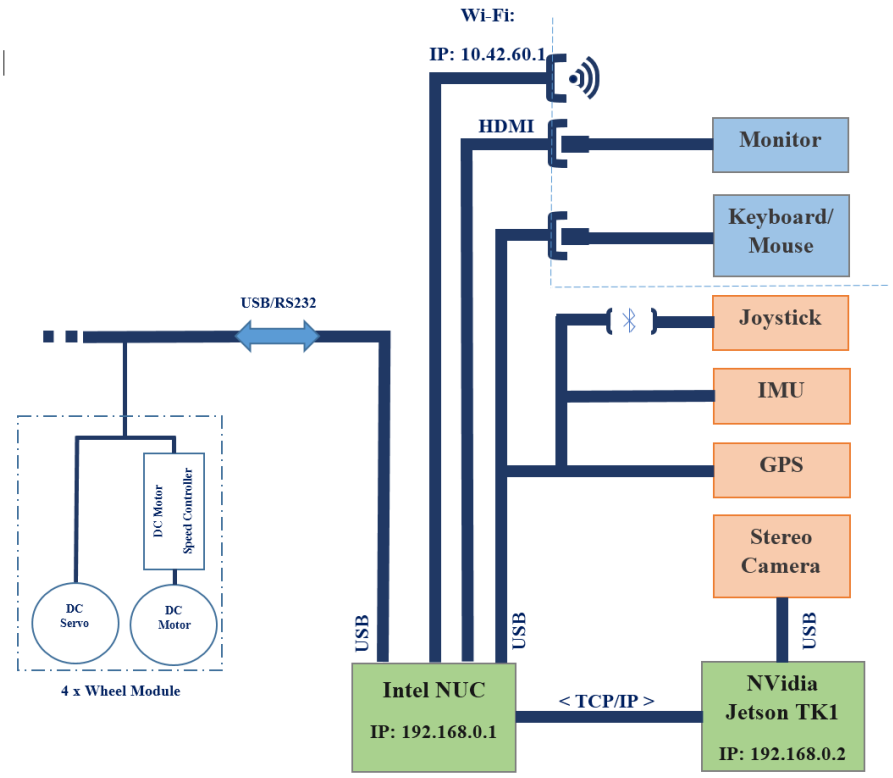


Figure 3.7: Component diagram of MARIO.

Unit (GPU), Nvidia Jetson Tegra TK1 is employed for this purpose. The ZED stereo camera is interfaced to the Jetson Tegra TK1 through USB3 connection. Jetson Tegra TK1 as a unit is connected to the NUC through TCP/IP communication.

The IMU is used to provides information about the relative rotation and acceleration of the robot body while the GPS provides absolute position information. The stereo camera is used for vision based localization and navigation purposes. Three navigation scenarios have been considered for operation of this robot including, teleoperation, semi-autonomous and fully autonomous navigation systems. Teleoperation control is achieved using a PS3 joystick connected through the Bluetooth to NUC. Figure 3.8 shows the final prototype of MARIO.

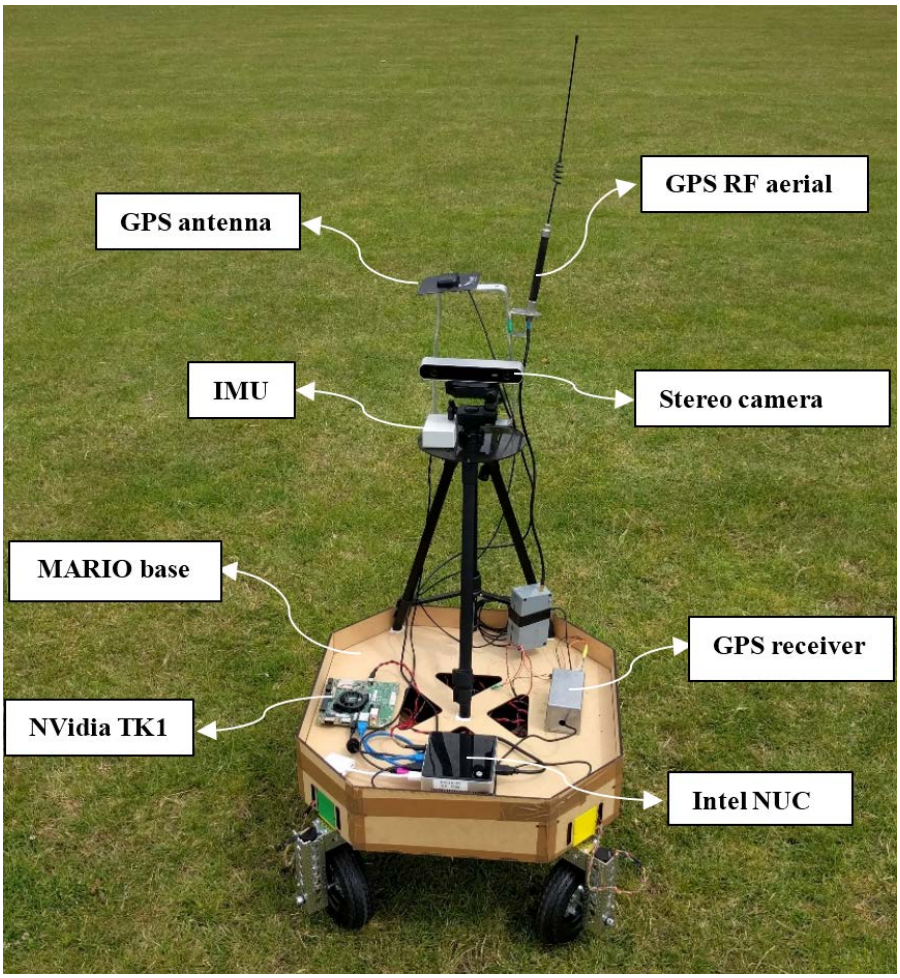


Figure 3.8: Final prototype of MARIO.

3.3.2 Software System

The software system development for MARIO includes 3 stages of modelling, simulation, and final implementation. Modelling and simulation enable software development, test and validation in a virtual environment. After successful development, the software system can be transferred to the physical system. Simulation has been carried out using Gazebo simulator which will be described with more details in Chapter 4. Software is developed in ROS and it can be interfaced to the physical or simulated model.

ROS is an open source robotic middle-ware which provides libraries and tools to help software developers produce robot programs. It provides hardware abstraction, device drivers,

libraries, visualizers, message-passing, package management, and more [41]. Robot programs for each of the modules are called ROS nodes. The protocol of message subscription-advertising enables a node, to subscribe the input data and publish the output data after processing once the node is executed.

ROS nodes can communicate with each other by passing the messages. A ROS message is a data structure, with a specific type that holds a specific topic name for identification. A node can subscribe to a topic that carries a message from another node if the specific topic and message type is defined properly for the subscriber. All the communications are handled by the ROS Master which enables each of the nodes to identify each other. The ROS publisher-/subscriber communication protocol is shown in Figure 3.9.

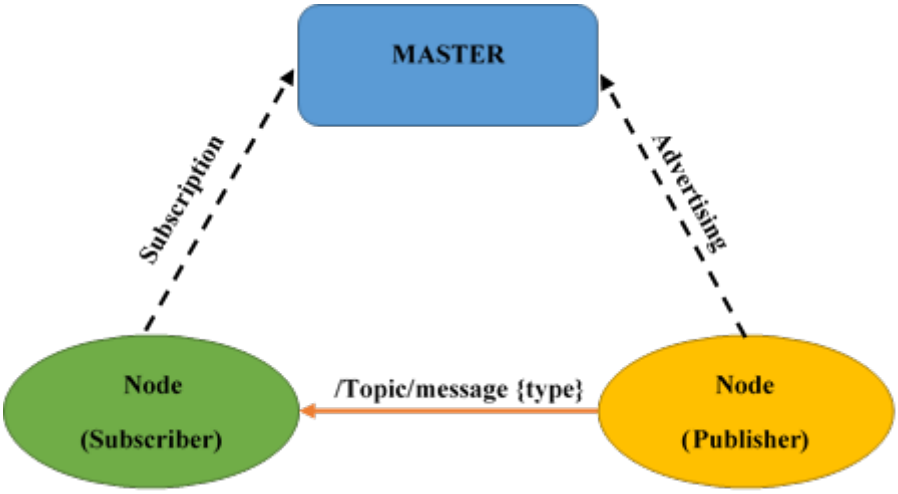


Figure 3.9: Node-Node and Node-Master interactions in ROS.

ROS nodes are programmed in C++ or Python programming languages. These programs are included in the ROS packages. The basic packages needed for MARIO to enable the control system and the sensors are shown in Figure 3.10. The represented control system includes *mario_hw_node* which is responsible for communicating with the actuators and *mario_robot_control* which runs the kinematic control system.

A kinematic-based controller (described in Chapter 4) is designed based on the kinematics of the system to drive the robot given the desired input linear and angular velocities for three types of steering including, spot turn, crab steering, and four wheels coordinated steering. This controller controls the desired speed of each of the four driving DC motors and steering servos through inverse kinematics. Rotation and translation of the robot with respect to the world frame is obtained through forward kinematics as wheel odometry.

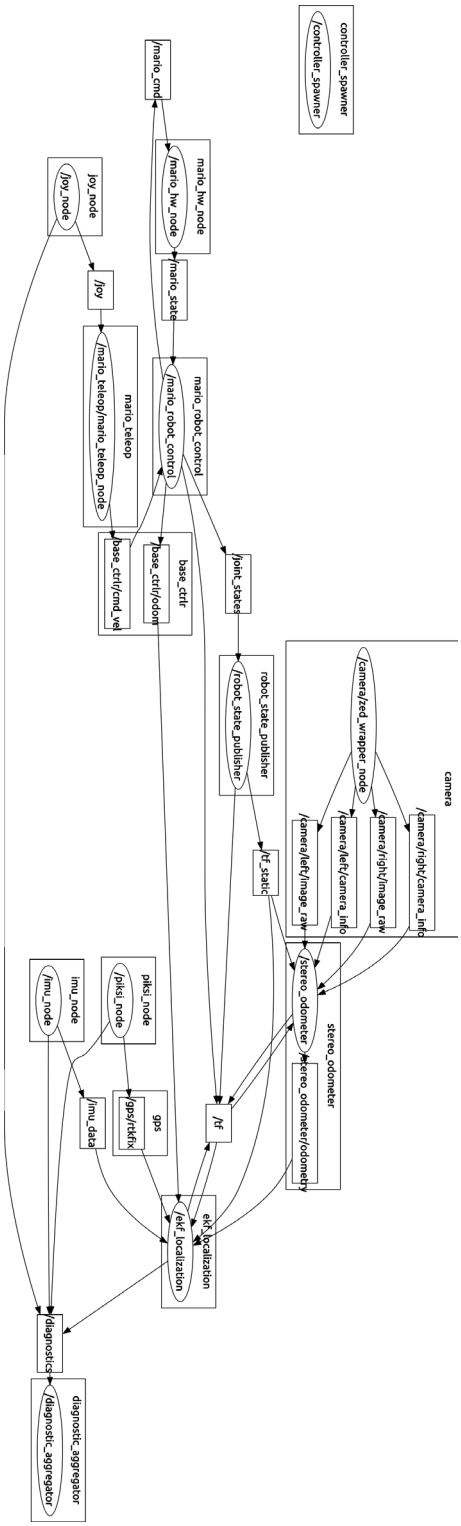


Figure 3.10: All active nodes and the associated topics for basic teleoperation of MARIO.

Teleoperation control of the robot is provided through the *joy_node* which is the driver of the joystick, and the *mario_teleop_node* which subscribes to the */joy* message and publishes */base_ctrlr/cmd_vel* as the input velocity to the MARIO control system. The ZED stereo camera, Piksi RTK GPS, and Razor IMU are activated using package drives represented as */camera/zed_wrapper_node*, */piksi_node*, and */imu_node* respectively.

ROS robot_localization package from the Navigation stack is used to fuse the localization data from the wheel odometry, stereo visual odometry, IMU, GPS for better and more accurate localization. For this aim, */ekf_localization* node receives the required topics from the sensors and fuses all the data using and *extended Kalman filter*.

Trajectory planner as the main part of the navigation system is achieved by the ROS *Navigation Stack*. 2D navigation from *Navigation Stack* takes pose information of the robot from the state estimation system (*/ekf_localization*) and a goal pose from user input. It generates a path from the current location of the robot to the goal location by its global planner, it then generates the safe velocity commands (*/base_ctrlr/cmd_vel*) sent to the robot base. This system drives the robot close to the generated path toward the goal point. Figure 3.11 shows visualization of MARIO in a simple 2D navigation situation in *Rviz*. *Rviz* is the ROS 3D visualization service that provides tools for live visualization of sensor data and state information from ROS.

3.4 Summary

In this chapter, a comprehensive mechatronic design and development procedure of the non-holonomic omnidirectional mobile robot - MARIO has been presented. To achieve this, an integrated application of CAD/CAE/CAM has been performed. This allowed us to develop a mobile robot which meets the design requirements such as modularity, assemblability and

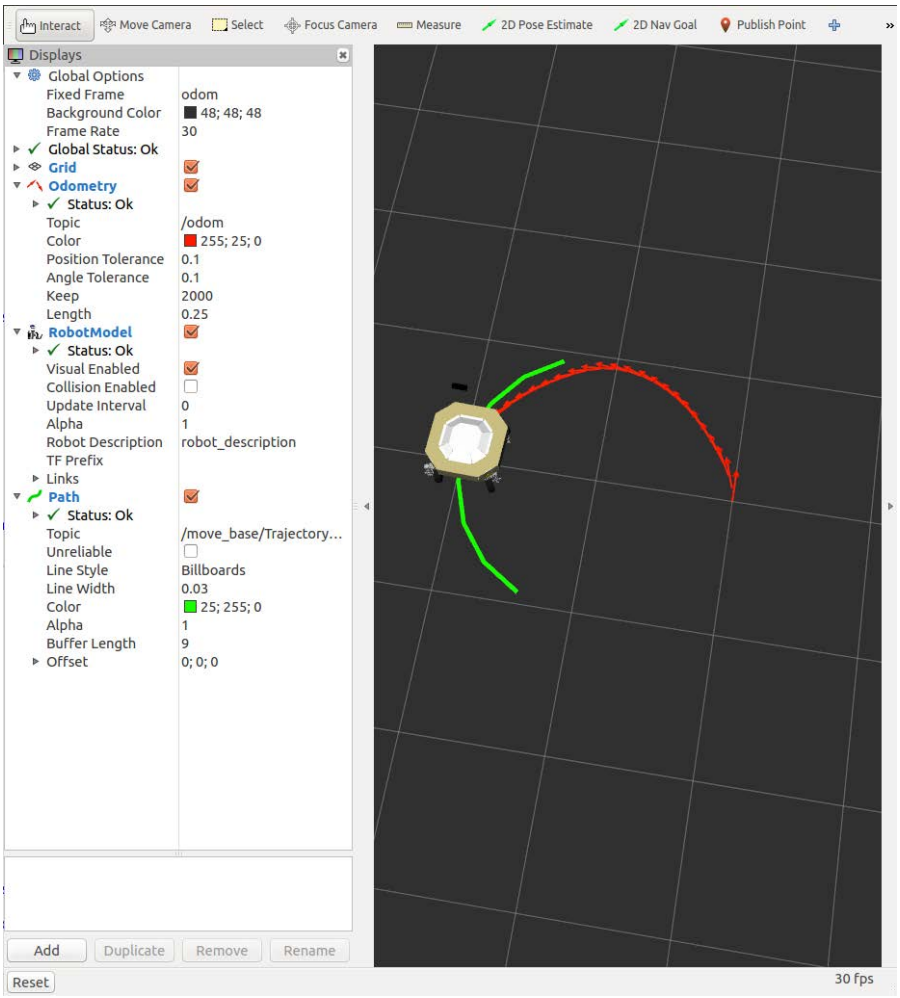


Figure 3.11: MARIO 2D navigation visualization in *Rviz*, green path: global path generated by the trajectory planner, red path: the travelled odometry from the robot state estimation system.

manufacturability. By this integrated application, MARIO has been developed successfully with a relatively low cost and time. MARIO has a low weight chassis and parts with adequate stiffness which weighs 10 kg in total. After developing electronics, control and navigation system, it drives stably with the maximum speed of 1.263 $^{m}/_{s}$. Kinematic modelling and simulation, control, and navigation will be described in the next chapters.

Chapter 4

Modelling and Simulation of Non-Holonomic Omnidirectional Robot

4.1 Introduction

This chapter presents 3D modelling and simulation of MARIO, using the Gazebo simulator and ROS, aiming for offline programming and system performance analysis. For this purpose, MARIO is modelled and simulated based on the physical developed model. Gazebo enables simulation of the world environment, physical model, sensors and control system through the URDF file. ROS is interfaced with Gazebo which allows utilization and implementation of different robotic software and tools on the simulated robot. This presented approach allows development, testing and validation of MARIO and required software before implementation on the real system.

4.2 Model Description and Simulation

In this section, all the physical links and joints, as well as kinematics and dynamics, sensing and basic control interfaces of MARIO are modelled and described individually. The Gazebo

simulator provides dynamics simulation by considering gravity, friction, and contact forces provided by the included physics engines such as Open Dynamics Engine - ODE, Simbody, Dynamic Animation and Robotics Toolkit - DART or Bullet. Different types of Gazebo plugins enable the development of control interfaces and sensing systems for the simulated robots. The main parts required to model a controllable robotic system in general are:

1. World environment model description
2. Physical model description
    - Kinematic and dynamic modelling of the robot links
    - Kinematic and dynamic modelling of the robot joints
3. ROS/Gazebo Plugins to model the sensors and control/hardware interfaces

Figure 4.1 shows the general structure and the required components in Gazebo to model a robotic system.

4.2.1 World Description

World is a general term to describe objects, global parameters and physics properties. By default, a world is defined by Gazebo with default required parameters. The objects in the world can be static or dynamic. Static objects such as building, lights or walls are defined by their visual and collision geometry. Dynamic objects such as robots are defined not only by visual and collision geometry but also by their inertia information. Objects can be created using standard geometric shapes, or inserted from the model database, or created and by any 3D modelling tools and imported to the Gazebo simulator environment.

vineyard development, including the terrain and the vine trees with parameters

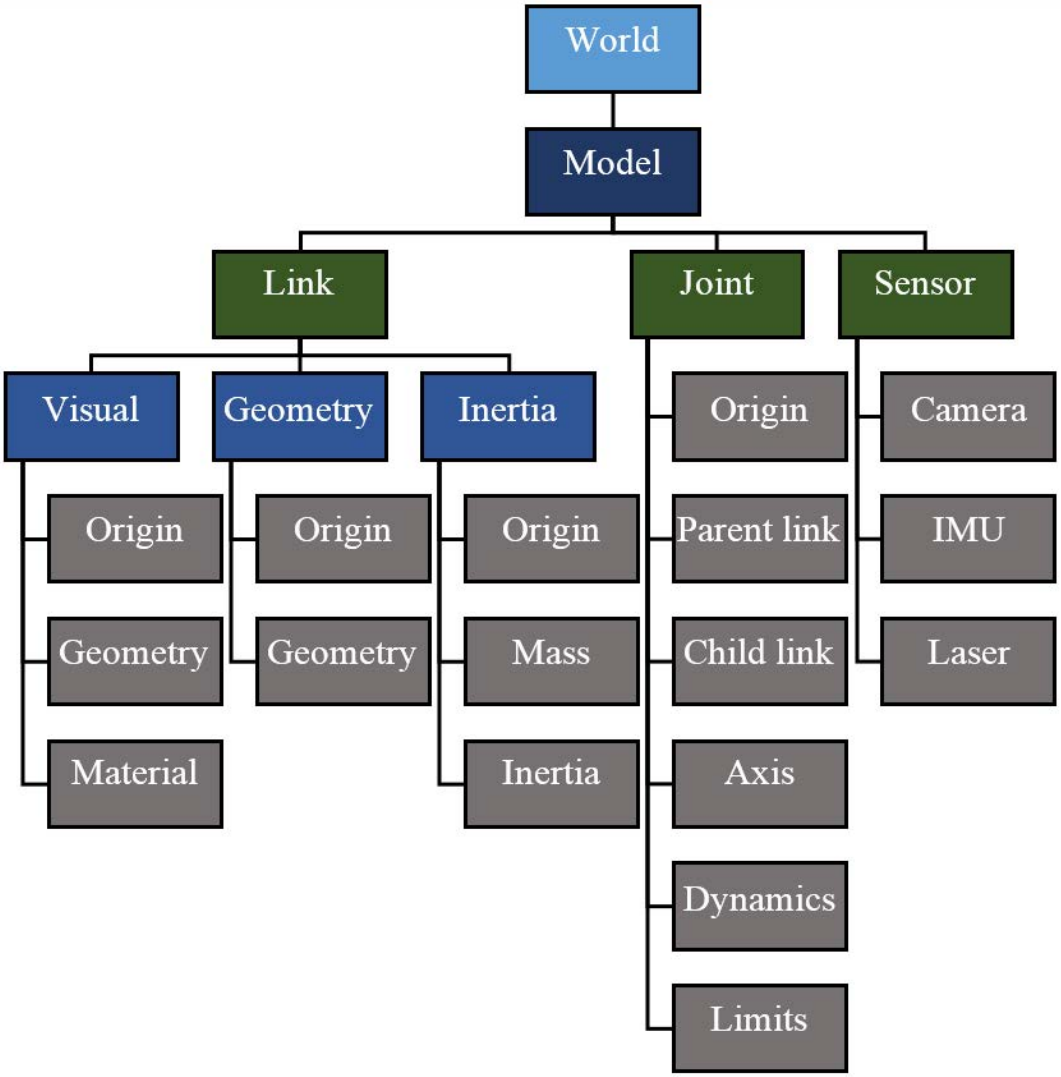


Figure 4.1: General structure of required components to model a robotic system in Gazebo.

Figure 4.2 shows a simulated environment in Gazebo with the set physics parameters such as gravity. Two static objects as the modelled terrain and agricultural environment (vineyard) are also included in the simulated world environment. Both objects were created using the SketchUp software.

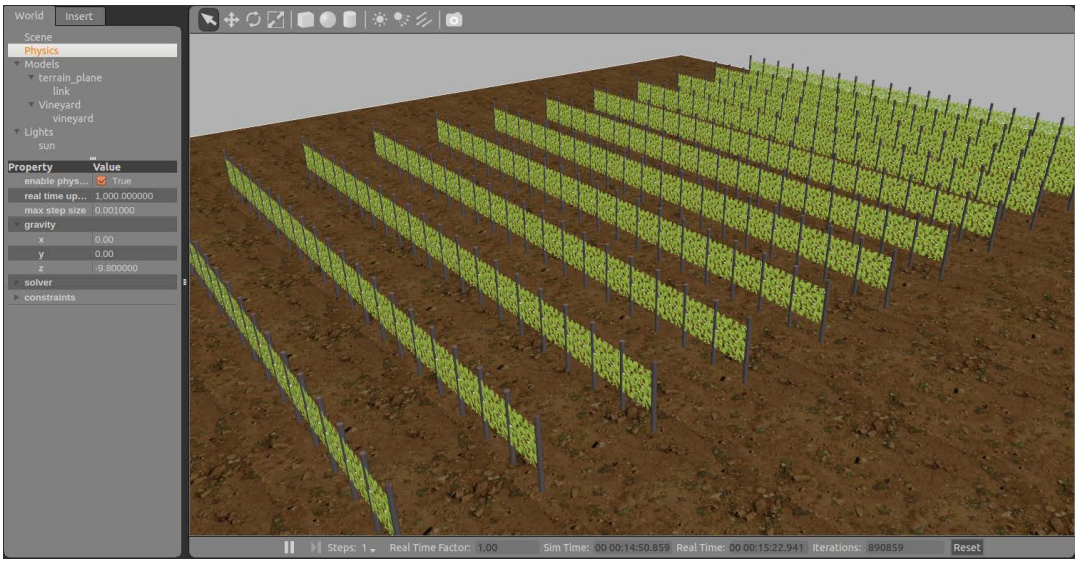


Figure 4.2: Gazebo simulator environment including objects in the world and world parameters.

4.2.2 Physical Model Description

A robot, as a dynamic object in the world, consists of links that are connected to each other by joints. A link in Gazebo describes the kinematic and dynamic properties of a physical link in the form of visual/collision geometry and inertia information. A joint models kinematic and dynamic properties of a joint such as joint type, motion axes, and joint safety limits. All these information are described in the Universal Robotic Description Format - URDF file format [40]. URDF is an XML file format used by Gazebo and ROS to model all the components of a robot. To model MARIO as a 4WD4S platform, 9 links and 8 joints need to be defined.

The base_link describes the chassis and four links (link1 to link4) are connected to it by four revolute joints (servo1 to servo4) as steering links. Each of the steering links are connected to a driving link (link11 to link44) as wheel by a continuous joint (motor1 to motor4). Each sensor also should be defined as a physical link attached with a fixed type joint to the base_link. with no attached sensor. A virtual link called base_footprint is also needed by some of the of the ROS third party software such as navigation which is placed on the ground and it is sized as the footprint of the platform. Figure 4.3 presents the kinematic schematics of the

base_link a wheel module with its links and joints. Figure 4.4 shows the kinematic diagram of MARIO platform, showing the kinematic chain of all the physical and virtual links and joints, each joint with the 6 DOFs information of its placement with respect to the previous link.

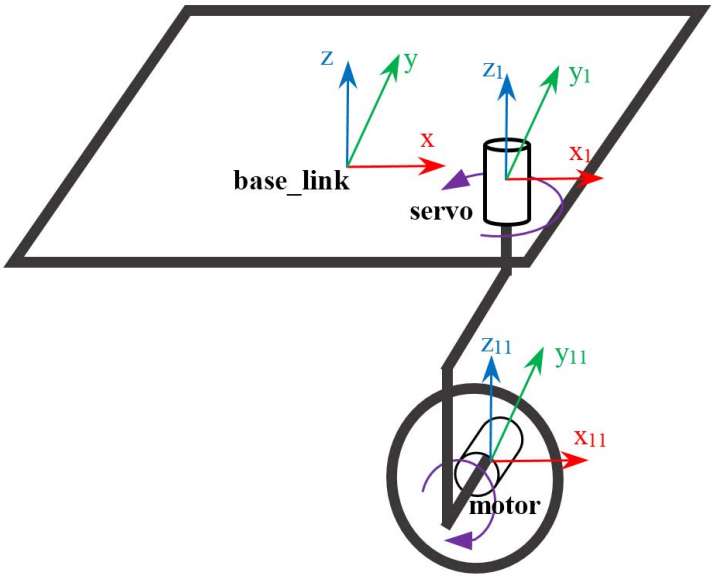


Figure 4.3: Kinematic diagram of MARIO: Kinematic schematic and coordinates of the base_link and one wheel module.

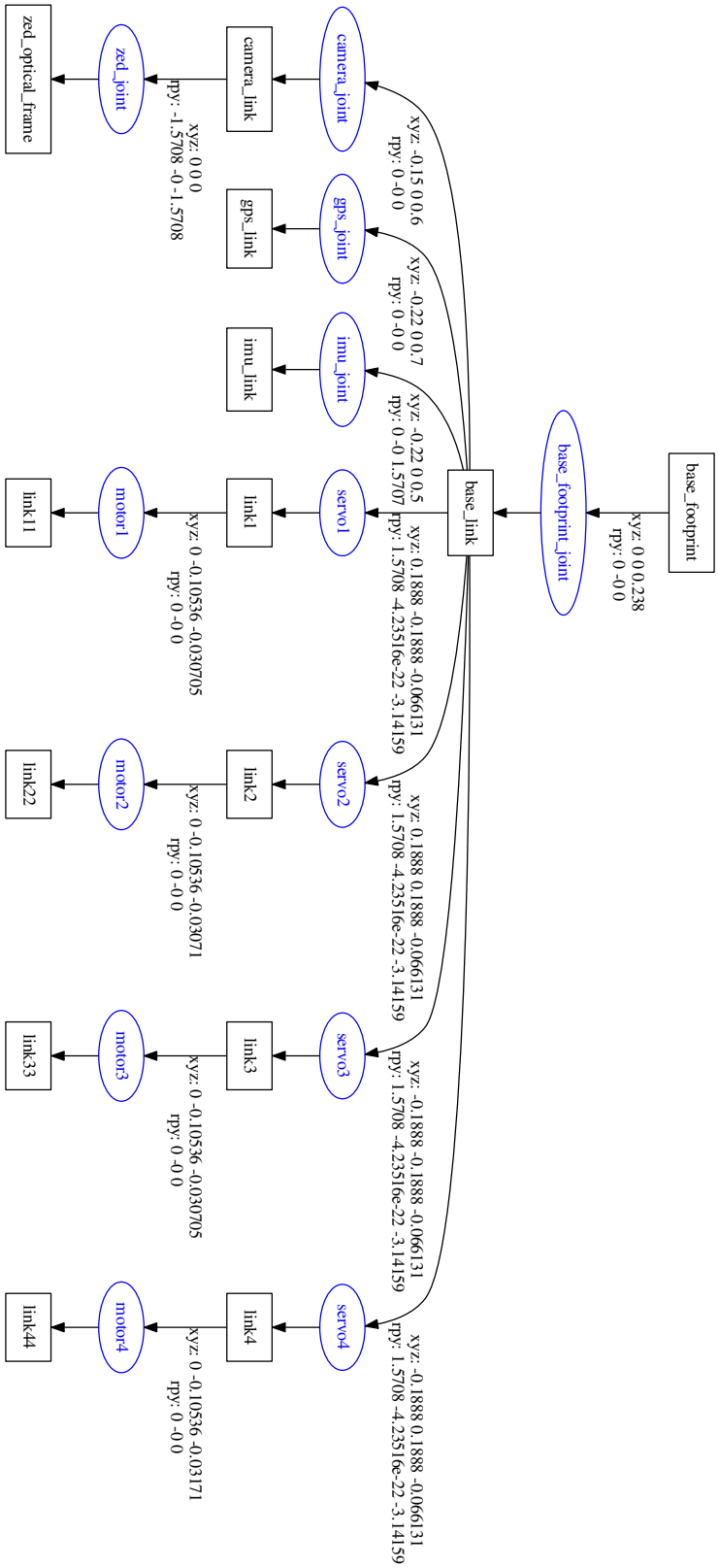


Figure 4.4: Kinematic diagram of MARIO: Kinematic chain generated from URDF file.

4.2.2.1 Physical Geometry and Inertia

The physical geometry of each link needs to be defined in the form of visual and collision geometry. There are two ways to model the geometry of each link of a robot, inserting standard 3D shapes, or importing as mesh files. Physical geometry for a typical four-wheel mobile robot can be modelled using a cubic box as the chassis and 4 cylindrical shapes as the wheels. Using continuous joints, wheels can be attached to the chassis. Inertial information of each link is essential, if a proper simulation is required. Inertial parameters define the mass, centre of the mass, and the moment of inertia tensor matrix for each modelled link. These information are obtained from SolidWorks. Below is the example of MARIO URDF code to model the imu_link.

```
1   <link name="imu_link">
2     <visual>
3       <geometry>
4         <box size="0.08 0.05 0.03"/>
5       </geometry>
6       <material name="blue">
7         <color rgba="0 0 .9 1"/>
8       </material>
9     </visual>
10     <collision>
11       <geometry>
12         <box size="0.08 0.05 0.03"/>
13       </geometry>
14     </collision>
15     <inertial>
16       <mass value="0.02"/>
17       <inertia ixx="0.0001" ixy="0" ixz="0" iyy="0.000001" iyz="0" izz="0.0001"/>
18     </inertial>
19   </link>
```

The alternative approach to define visual and collision geometry is to use mesh files in the format of COLLADA or STL files. These files can be generated by CAD tools such as Blender, SolidWorks or SketchUp. The MARIO 3D CAD model, Figure 3.1, was developed using

SolidWorks as described in the previous Chapter. To provide colour information, COLLADA file format can be used to present visual geometry of a link. Collision geometry does not need material or colour information, therefore it can be represented as STL file format.

Joint Representation Joints are needed to connect the links to each other and form the kinematic and dynamic relationships between them. Joint element in the URDF file defines the kinematics and dynamics of the joint as well as joint type and safety limits. Figure 4.5 shows the tree structure of a two-link robot for better understanding of the different terms in joint element. Below is the modelled continuous joint element that connects the steering link (link1) to the driving link (link11) in MARIO.

```
1  <joint name="motor1" type="continuous">
2     xyz="0.0 -0.10536 -0.030705"     rpy="0.0 0.0 0.0" />
3     <parent link="link1"/>
4     <child link="link11"/>
5     <axis xyz="0 0 1" />
6     <dynamics damping="0.0" friction="0.0"/>
7     <limit effort="30" velocity="2.0"/>
8  </joint>
```

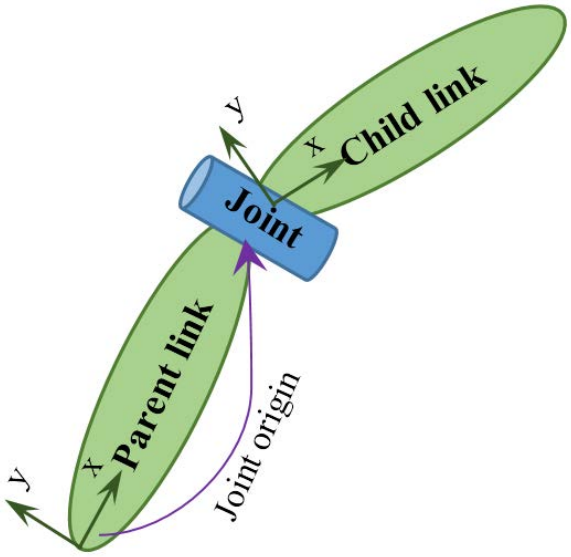


Figure 4.5: Tree structure of a two-link robot.

4.2.3 Sensor Modelling

With many different sensors on the robot, each sensor can be simulated independently as a Gazebo plugin and must be physically attached to the robot model as a link. These plugins are included in the URDF file. These plugins are typically C++ libraries loaded by Gazebo that have access to Gazebo's API. In general, plugins are permitted by Gazebo to perform different kinds of task such as motion control or getting sensor data. These plugins output sensor information in form of standard ROS messages and services. Common parameters, such as error characteristics and transformation frame of each sensor, can be defined as well as other parameters related to each sensor. By default, sensors in Gazebo have no noise and they sense the simulated environment perfectly. To make them more realistic, noise can be added. A first order Gaussian error model is typically used for all the sensors to add noise to the measurement taken from each sensor. This is modelled by setting the mean and the standard deviation of the Gaussian distribution. Each measurement of $Y(t)$ at time t is given by:

$$Y = \hat{Y} + B + N_Y \tag{4.1}$$

$$\dot{B} = -\frac{1}{\tau}B + N_B \tag{4.2}$$

where in Equations 4.1 and 4.2, $\hat{Y}$ is the raw measured value, B is the bias, N_Y is additive noise that affects the measurement, and N_B defines the characteristics of random drift with time constant τ [67].

4.2.3.1 Inertial Measurement Unit (IMU)

The inertial measurement unit (IMU) reports the robot body's acceleration from a three axis accelerometer, angular rates from a three axis gyroscope and absolute orientation around the

Z axis by a magnetometer. Integration of these measurements with other sensors provides a good reference for a localization system. Noise and bias are the two types of disturbance that are applied to the angular rates and acceleration measurements of IMU. Four groups of parameters need to be set for the IMU model: angular rate noise and bias, acceleration noise and bias. Also, no noise or bias is added to the orientation measurement as it is considered a perfect value in the world frame. Noise is sampled and added from a Gaussian distribution by setting the mean and standard deviation of the Gaussian distribution. Bias is sampled once and is added at the start of simulation. The physical used IMU on MARIO is modelled with no associated noise in Gazebo, the code below shows the required URDF code to model the IMU.

```
1  <joint name="imu_joint" type="fixed">
2      <axis xyz="1 0 0"/>
3      <origin xyz="-0.16 -0.05 0.51"/>
4      <parent link="base_link"/>
5      <child link="imu_link"/>
6   </joint>
7    <gazebo>
8       <plugin name="imu_plugin" filename="libgazebo_ros_imu.so">
9           <alwaysOn>true</alwaysOn>
10           <bodyName>imu_link</bodyName>
11           <topicName>imu_data</topicName>
12           <serviceName>imu_service</serviceName>
13           <updateRate>10.0</updateRate>
14        </plugin>
15    </gazebo>
```

4.2.3.2 Stereo Vision Camera

Stereo camera is a vision system using at least two cameras to simulate the way human vision works. Stereo camera gives the ability to generate 3D images. Typically, two cameras that are

placed horizontally from each other are used to capture images from different angles. Through several pre-processing steps, depth information is achieved as a disparity map. The disparity map includes the information of the differences in horizontal coordinates of the two input images. The Multi-camera plugin used by Gazebo simulates a stereo camera and synchronises their output. This plugin is similar to the Camera plugin that simulates a monocular camera. All the required parameters such as frame rate, image width and height, output format, base line distance and noise parameters can be defined within Gazebo. A Gaussian disturbance is sampled for each pixel individually and it is added to each colour channel of that pixel. In our system, a stereo vision system is used as a visual odometry source for the state estimation system to enhance the localization of the robot. Figure 4.6 shows the visualization of simulated stereo camera by showing the left and right images of an example scene in Gazebo. The parameters from the physical ZED stereo camera were used to model the simulated stereo camera.

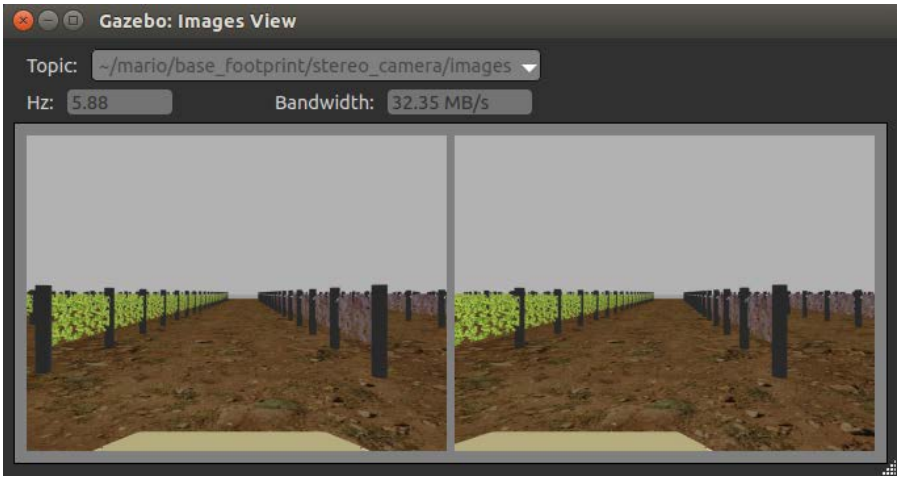


Figure 4.6: Visualization of the simulated stereo camera as left and right images.

4.2.4 Control plugin

To be able to control the motion of each joint in one degree of freedom, a Gazebo control plugin is required. Gazebo also needs to be interfaced with a robot middle-ware such as ROS to control each joint. A meta package called *gazebo_ros_pkgs* provides a set of ROS packages to interface with Gazebo. A set of packages called *ros_control* provide controllers, hardware interfaces and toolboxes to control joint actuators. Common controllers such as *effort*, *position* and *velocity* are provided by *ros_control*. These controllers (typically PID type) take joint states and set points as inputs and output effort, position or velocity. Each joint is interfaced with the relevant controller by the relevant hardware interface. The same developed software control system is used to control the physical robot, so that the control messages are written to the physical speed/position controllers to actuate control the actuation of the joints. The speed and position feedbacks are read from the encoders back to the control system.

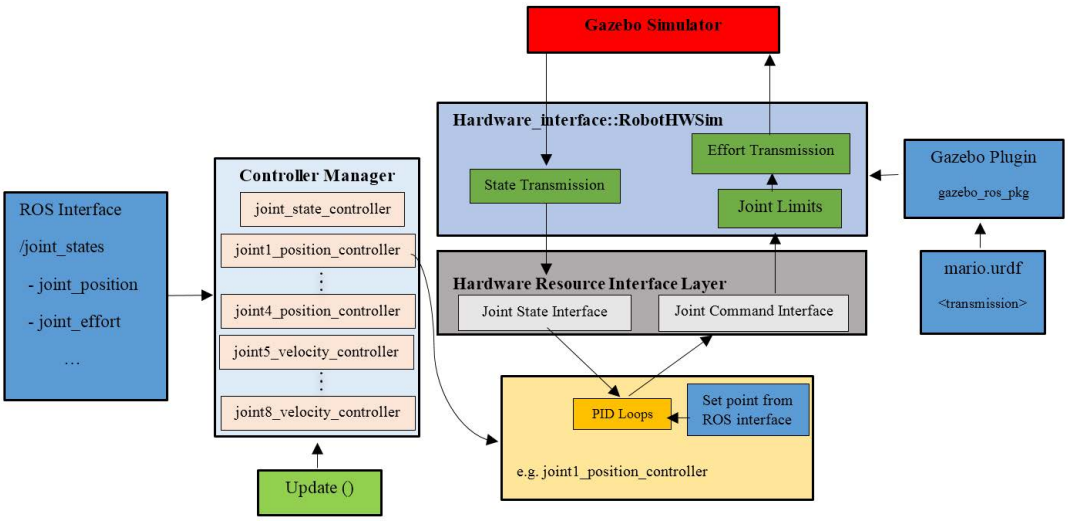


Figure 4.7: Overview chart of low level control system of simulated MARIO in Gazebo and ROS.

An overview of the low level control system of simulated MARIO is shown in Figure 4.7. By launching the ROS package containing the URDF file of MARIO, the simulated model is

opened in the virtual world of Gazebo. This also loads the control interface and waits for all the controllers to be loaded. The transmission tag in the MARIO URDF file defines the type of command interface and the relationship between the joint and actuator. Controller manager provides the infrastructure to load, start, stop and unload the controllers in a real-time manner. A YAML configuration file is also needed which includes information of controllers such as joint controller type and parameters. Hardware_interface provides position, velocity and effort interfaces between ROS and Gazebo. Our system has 9 controllers of which one provides joint states, four are position controllers to control steering joints and remaining four are velocity controllers to control each of the driving joints. Figure 4.8 shows the performance of the PID position controller of joint1 and the response of the controller to the step input from 0.5 radians to 0.

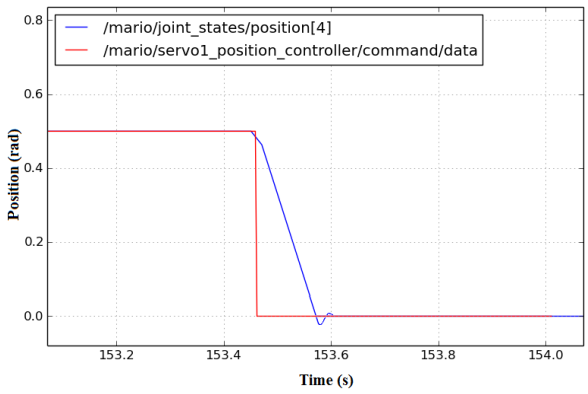


Figure 4.8: Joint position controller performance.

4.2.5 Standard Kinematics Model

In general, ros_control provides a closed loop control system for each individual joint. To model a robotic system in Gazebo, kinematic formulation of the system is not required. However, kinematic formulation is essential to develop the control system. The kinematic diagram

of MARIO is shown in Figure 4.9. The robot frame is assumed as a rigid body that moves in a planar motion.

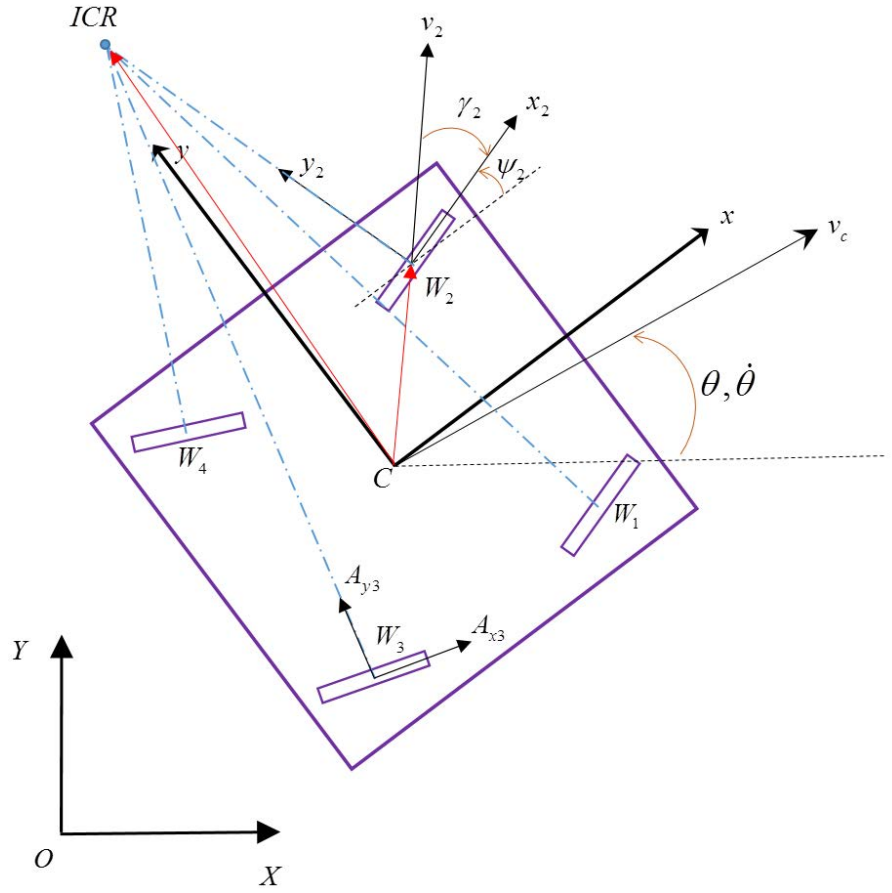


Figure 4.9: Kinematic notation of MARIO.

Velocity of each of the four wheels can be formulated as:

$$\overrightarrow{v_i} = \overrightarrow{v_c} + \dot{\theta} * k * \overrightarrow{W_i} \tag{4.3}$$

where in Equation 4.3, $\overrightarrow{v_i}$ is the speed vector of each wheels, $\overrightarrow{v_c}$ is the linear speed vector and $\dot{\theta}$ is the angular rate of the robot's frame in $\{C, x, y\}$coordinate with respect to the fixed global coordinate $\{O, X, Y\}$, $\overrightarrow{W_i}$ is the distance vector of each wheel from the centre of robot's frame, and k is a unit vertical vector.

In the pure rolling condition, the wheel rotation rate, $\dot{\phi}_i$ and steering angle, ψ_i for a wheel with the radius of r with respect to the robot's frame coordinate system are given by:

$$\dot{\phi}_i = \frac{|v_i|}{r}, \quad tan(\psi_i) = \frac{v_{iy}}{v_{ix}}, \quad i = 1, ..., 4, \tag{4.4}$$

where in Equation 4.4, v_{ix} and v_{iy} are the vector components of $\overrightarrow{v_i}$ in the robot's local frame$\{C, x, y\}$ as:

$$v_{ix} = v_{cx} - W_{iy}\dot{\theta}, \quad v_{iy} = v_{cy} - W_{ix}\dot{\theta}, \tag{4.5}$$

The two terms in Equation 4.4 define the kinematics constraints of driving speeds and steering angles of wheels with respect to the linear and angular velocities at the robot's frame. Based on Equation 4.4, the slip angle of each wheel γ_i can be formulated as:

$$\gamma_i = cos^{-1}\left(\frac{v_{ix}}{|v_i|}\right), \tag{4.6}$$

Non-zero input linear and angular velocities to the base result the whole platform to rotate around a centre of rotation based on the concept of rotational motion of rigid body. The instantaneous centre of rotation ($\overrightarrow{ICR}$) based on the input velocity is formulated as:

$$\overrightarrow{ICR} = \frac{\vec{v}_c}{\dot{\theta}}, \tag{4.7}$$

Acceleration of each wheel a_i at point w_i and the acceleration of robot's frame a_c at point C based on the kinematics are given by:

$$a_i = a_c + \ddot{\theta} k\overrightarrow{W_i} - \dot{\theta}^2\overrightarrow{W_i}, \tag{4.8}$$

where

$$a_c + a_{cx} + a_{cy}, \quad a_{cx} = \left(\dot{v}_x - \dot{\theta} v_y\right), \quad a_{cy} = \left(\dot{v}_y + \dot{\theta} v_x\right), \tag{4.9}$$

and by knowing $a_i = a_{cx} + a_{cy}$, therefore:

$$a_{ix} = \dot{v}_x - \dot{\theta} v_y - W_{ix}\dot{\theta}^2 - W_{iy}\ddot{\theta}, \tag{4.10}$$

$$a_{iy} = \dot{v}_y + \dot{\theta} v_x - W_{iy}\dot{\theta}^2 + W_{ix}\ddot{\theta}, \tag{4.11}$$

4.3 State Estimation, Control and Navigation by ROS

4.3.1 Model Base Controller (MBC)

A model based controller (MBC) based on the kinematics of the system is responsible to drive the robot with a desired input speed. Through inverse kinematics, desired speed information in the local frame in the form of linear and angular velocities is received and the controller outputs driving speed and steering angle for each wheel. These outputs are set points for the relevant ROS controllers to control the position, velocity, or effort on each joint. The formulation provided in Equations (4.3 to 4.7) used in the process of inverse kinematics. The MBC is designed to switch between 4 modes of steering based on the received input velocity command as follows:

- Angular and linear velocity is zero results in a stop command for the wheels;
- Angular velocity is zero and the linear velocity is non-zero. In this case the wheels are aligned with the linear velocity vector of the base;
- Angular velocity is non-zero and the linear velocity is zero. In this case the robot ex-

ecutes a zero turn, where the wheels are angled tangential to the robots centroid.

- Rotational and linear velocity is non-zero. In this case the centre of rotation is calculated (Equation 4.7) and the wheels are angled tangential to to that centre.

The forward kinematics in MBC is achieved using the Kabsch algorithm [53]. This algorithm computes the best fit translation between two related sets of points $(\{P,Q\})$ based on the minimization of the Root Mean Square Deviations (RMSD). The algorithm works by calculating the covariance matrix of the related sets of point $A = P^T Q$. Using this covariance matrix the rotation matrix is calculated using a Singular Value Decomposition (SVD) depicted in Equations 4.12 and 4.13.

$$A = VSW^T, \tag{4.12}$$

$$R = W \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & d \end{bmatrix} V^T, \tag{4.13}$$

where R is the resulting rotation matrix and $d = sign(det(WV^T))$ in order to preserve a right-handed coordinate system. This rotation matrix is then used to to calculate the transformation between the sets of points. In terms of the odometry information each point in the set is the old position of each wheel and the new position of each wheel. The calculated transformation then approximates the relative motion of the base.

The kinematic calculations in the MBC are implemented in the robot as part of the propulsion modules. In practice the combination of linear and angular velocity commands prove difficult to follow, especially where the resulting ICR is small. In fact, when the radius of the ICR approaches the edge of the casing, the system barley achieves motion. This is caused by the disparity between the mathematical representation of the robot compared to reality. In the mathematical model, the wheels are represented as a point in space whereas, the wheels on the

real robot have a rolling surface. This rolling surface causes friction in the system when the robot attempts to make a tight turn. Also ICR might be placed on the rotating joint and causes singularity which leads to joint and actuator damage. The ICR formulation and associated issues and solution are described in detail in the next chapter.

4.3.2 State estimation and Navigation

State estimation, control, and navigation systems are key components of a mobile robotic system to enable the robot to perform required tasks. Figure 4.10 shows an overview chart of the control system, state estimation and navigation system.

The state estimation system includes an EKF, using *robot_localization* package to estimate the robot pose by fusing all the measurements from multiple sensors. For our system, relative displacement in form of X, Y, and Z positions in the world frame is provided by wheel odometry and visual odometry. Orientation information is provided by IMU in the form of absolute orientations (roll, pith, and yaw) and angular velocities. In a 2D navigation scenario, yaw and Z axis angular velocity are needed to provide orientation information for EKF. EKF formulation and implementation is described in Chapter 6.

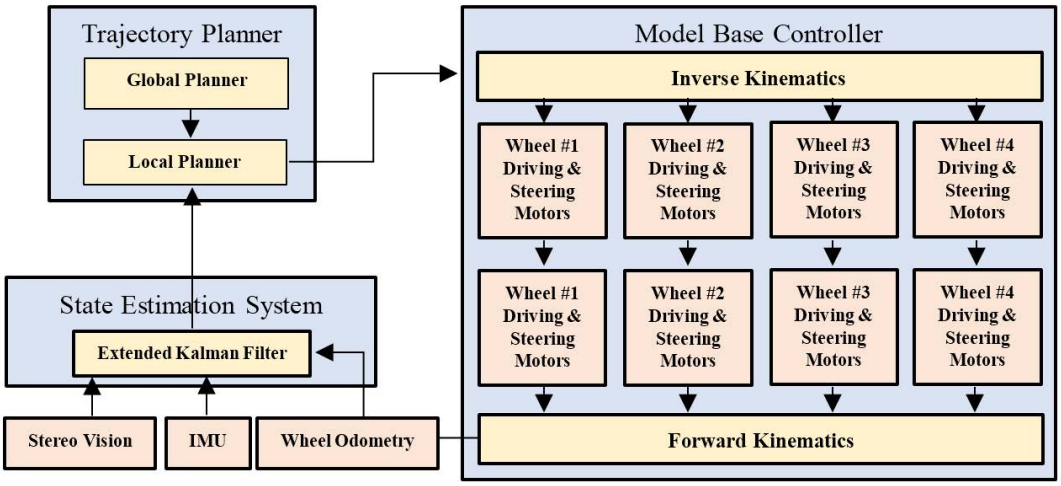


Figure 4.10: Overview chart of Control, state estimation and navigation system.

To drive the robot in the world frame, a navigation system is needed. This is achieved by the 2D navigation stack provided by ROS. Navigation stack takes pose information of the robot from the state estimation system and a goal pose from user input. It generates a path from the current location of the robot to the goal location by its global planner, it then generates the safe velocity commands sent to the robot base. This system drives the robot close to the generated path toward the goal point while avoiding obstacles.

4.4 Experimental Results and Discussions

The section provides the experimental results and discussions aiming for the validation of MBC and the performance of ROS third party software used for the state estimation and navigation systems. For each scenario, the input velocity command from a joystick has been recorded and then played back for the both simulated and physical MARIO. The results from each experiment were recorded for post processing and further analysis. Record/playback of the information is provided by *rosbag* tools in ROS.

4.4.1 Validation of the Kinematic Model and MBC

To validate the kinematic model and MBC, both real and simulated mobile robots perform a test trajectory by receiving the recorded velocity command from a joystick. Figure 4.11 shows the graphs of linear and angular velocity feedbacks for both systems received from the MBC in form of odometry (through forward kinematics).

The measured velocities from both simulated and real models present the similar behaviours in phase and magnitude, however the lag can be observed in the measured data from the

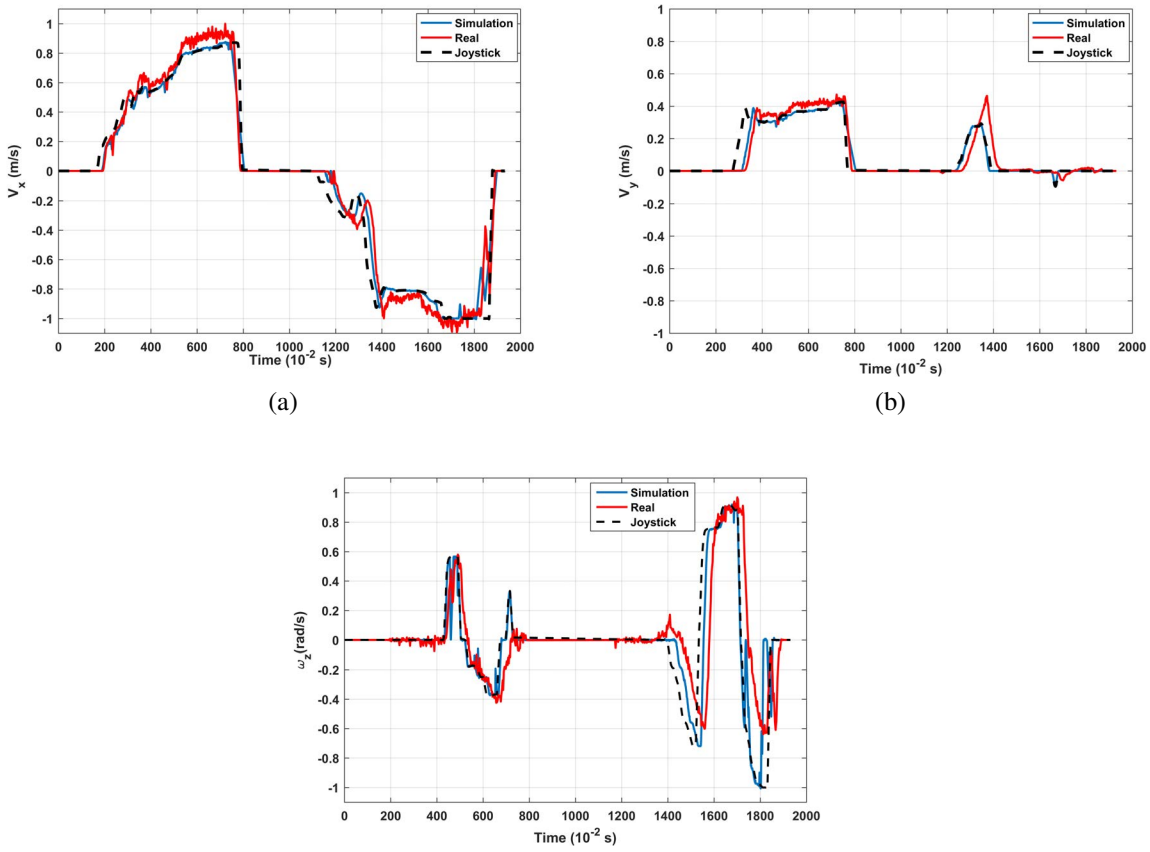


Figure 4.11: Graphs of measured linear and angular velocity values from the simulation, real system and velocity command: (a) Robot frame linear velocity v_x, (b) Robot frame linear velocity v_y , (c) Robot frame angular velocity $\dot{\theta}$.

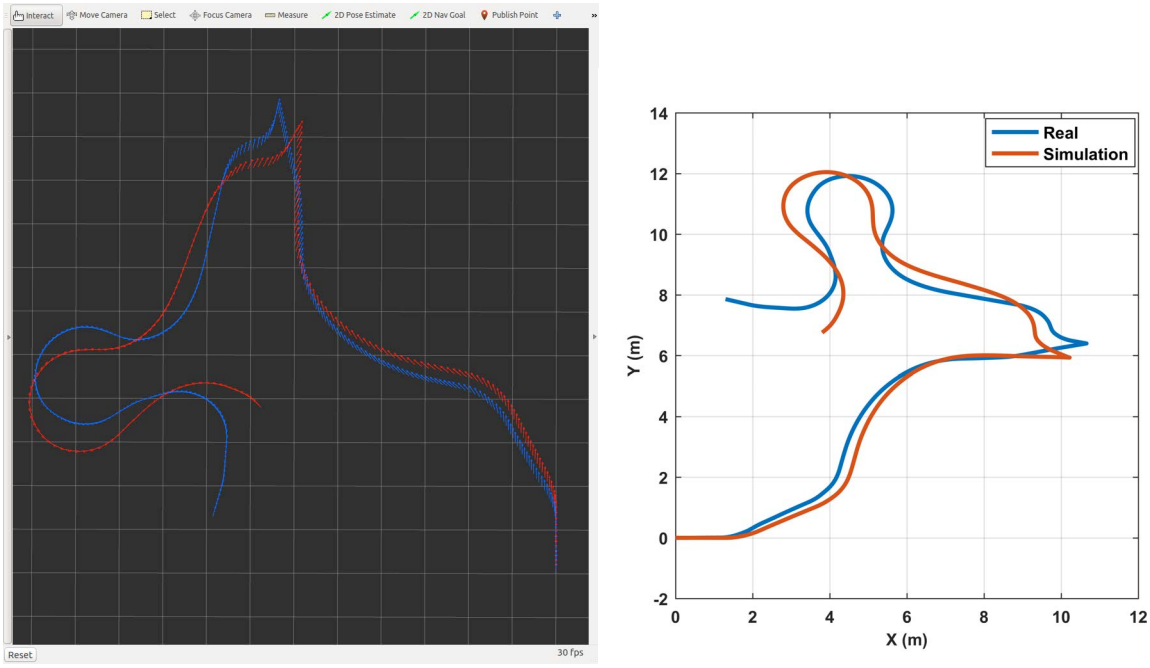
real model. This lag is due to the physical on-board speed controller response and the latency in the serial communication for transmitting the data from the main on-board computer to the physical DC motor controllers. The average percentage error for each axis velocity compared to the input velocity has been quantified and presented in Table 4.1. Although the error from the real model is almost the double of the simulation mainly due to the phase lag, but the experimental results is satisfactory for the performance validation of MBC.

Figure 4.12 demonstrates the travelled trajectory of the real and simulated robots by re-

Table 4.1: Comparison of the errors of both simulation and real model for the input velocity commands.

	Simulation error %	Real error %
v_x	4.36	7.38
v_y	1.39	3.34
$\dot{\theta}$	5.99	13.29

ceiving the same velocity command visualized by Rviz tool (ROS visualization tool). It was aimed to have all the possible combinations of $v_{x,}$, v_y, and $\dot{\theta}$generated by the joystick for this experiment.

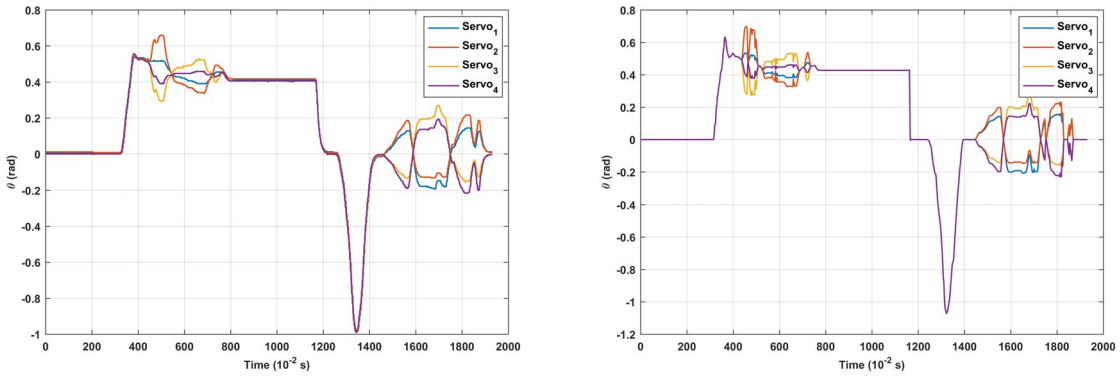


(a) (b)

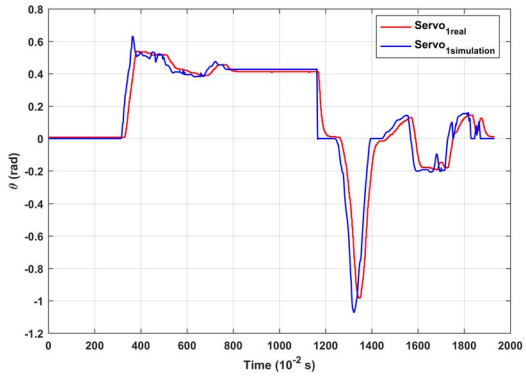
Figure 4.12: Travelled trajectory of both simulated and real MARIO on the same input velocity command: a) Rviz visualization, b) Labeled plot with axis-units.

To present the performance of MBC inverse kinematics, a set of results are presented in Figure 4.13 showing the performed steering control for each of the 4 servos during the experiment. Figure 4.13a depicts the results from the real model, while the simulation results

are provided in Figure 4.13b. The comparison between the real and simulation model is shown in 4.13c for the performed steering angle of servo1, as the phase lag is seen for the same reason described earlier. A cross correlation between the two signals quantifies 190ms phase lag, where the simulation leads the real. The presented results validates the performance of the MBC inverse kinematics.



(a) (b)

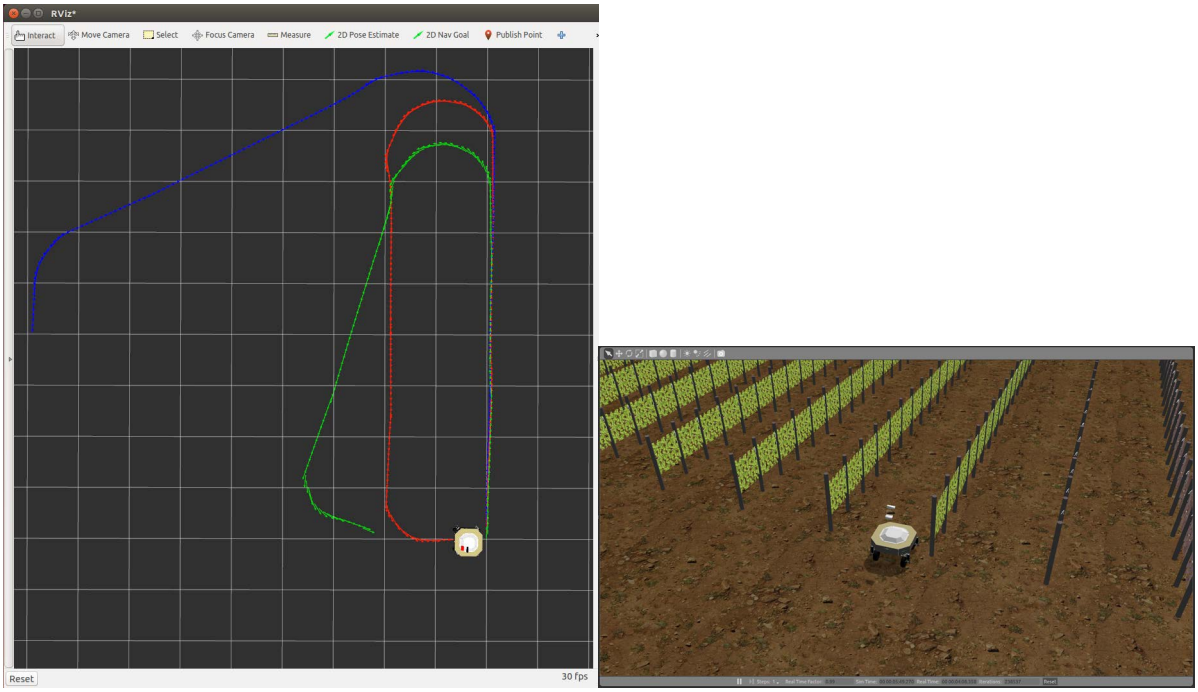


(c)

Figure 4.13: MBC inverse kinematics performance analysis: a) servos steering angles in real, b) servos steering angles in simulation, c) servo1 steering angles in both real and simulation.

4.4.2 State Estimation System Test

In this test scenario, simulated MARIO has been commanded using a joystick. A ROS teleoperation package has been used for this purpose to generate the velocity command for MARIO using a PS3 joystick. The purpose of this experiment is to test the performance of the state estimation system with the information from modelled sensors including the stereo camera and IMU in the simulation environment. Visual odometry is achieved through feeding the left and right images from the simulated stereo camera into *viso2_ros* package. *Viso2_ros* package from *libviso* library [43], is a feature based visual odometry program that outputs odometry measurements from a monocular or stereo camera. Visual odometry, wheel odometry, and IMU data have been integrated by the *EKF* from *robot_localization* package.



(a) (b)

Figure 4.14: Teleoperated navigation: (a) RViz visualization of odometry information for the travelled trajectory by MARIO, (b) The simulated vineyard environment for MARIO teleoperation in Gazebo.

Figure 4.14 shows MARIO in the simulated vineyard environment in Gazebo (Figure 4.14b), and the travelled trajectory visualized in RViz (Figure 4.14a), where the blue, green, and red plots represent the wheel odometry, visual odometry, and filtered odometry respectively. The filtered odometry is obtained from the *EKF*, where it fuses the IMU rotational information with both wheel and visual odometry displacement information. Both Wheel and visual odometry suffer from the accumulated drift over time. This drift is more presented in turns and it can be seen in Figure 4.14b. The results from the state estimation system show a better estimation of the travelled trajectory by the fusion of wheel odometry, visual odometry, and IMU data so that the robot is back to the home position (starting point) with the least drift ($0.3\,m$) compared to the wheel and visual odometry. Further analysis for comparison was not achievable, as the simulation of a GPS to provide the ground truth positioning information was not possible in Gazebo.

Figure 4.15 shows all the ROS nodes and topics involved in this experiment generated by the ROS command *rqt_graph*.

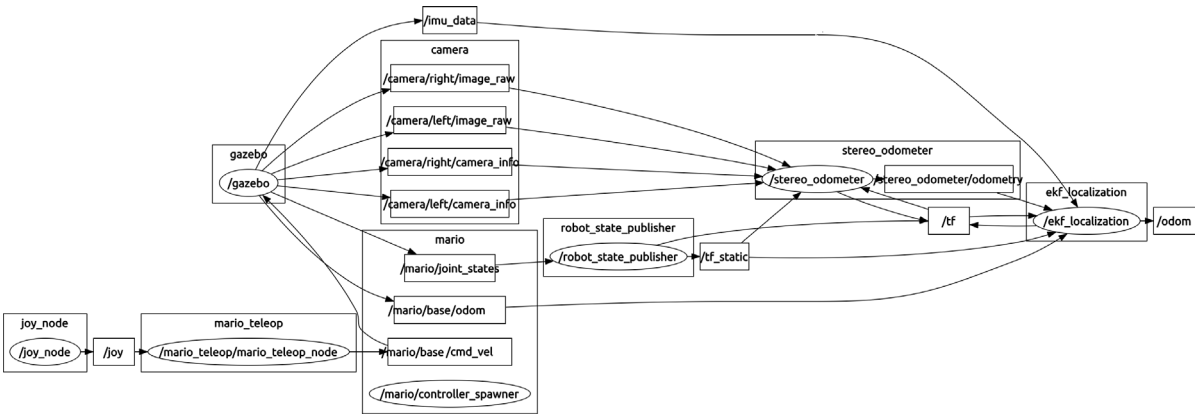


Figure 4.15: All the active ROS nodes (ellipses) and message topics (rectangles) flowing between them obtained from rqt_graph.

4.4.3 Semi-Autonomous 2D Navigation

In this scenario, 2D navigation has been performed to evaluate the performance of the navigation system as well as the sensory system. For this aim, the 2D navigation from ROS navigation stack has been used to command MARIO based on the planned trajectory. As shown in Figure 4.10 previously, the trajectory planner is composed of the global and local planners. The global planner plans and generates the trajectory in the local map, while local planner receives the current updated state of the robot from the state estimation system (*EKF*) and generates the velocity command for MARIO to drive the base on the desired trajectory. Figure 4.16 shows visualization of MARIO in a 2D navigation situation in *Rviz*.

4.5 Summary

This chapter has presented the modelling and simulation of a MARIO, using Gazebo simulator and ROS for the purpose of offline programming and system design validation. Gazebo simulator allowed modelling and simulation of different components of MARIO including physical model, sensing and control system. It also enabled the simulation of the world environment for robot operation. Interfacing Gazebo and ROS allowed access to a wide range of different robotic tools and software to be utilized in the simulated model. ROS provides the framework to develop the control, state estimation and navigation system. The kinematic model base controller was tested and validated successfully using the experimental results achieved in the simulation and real environment. Furthermore, the example scenarios of Gazebo/ROS for teleoperation control and semi-autonomous navigation of MARIO have been presented. Through this example, the performance of the simulated sensing system alongside the ROS state estimation and navigation system were tested and analysed. The approach provided in this chapter allows successful development and test of MARIO and different implemented

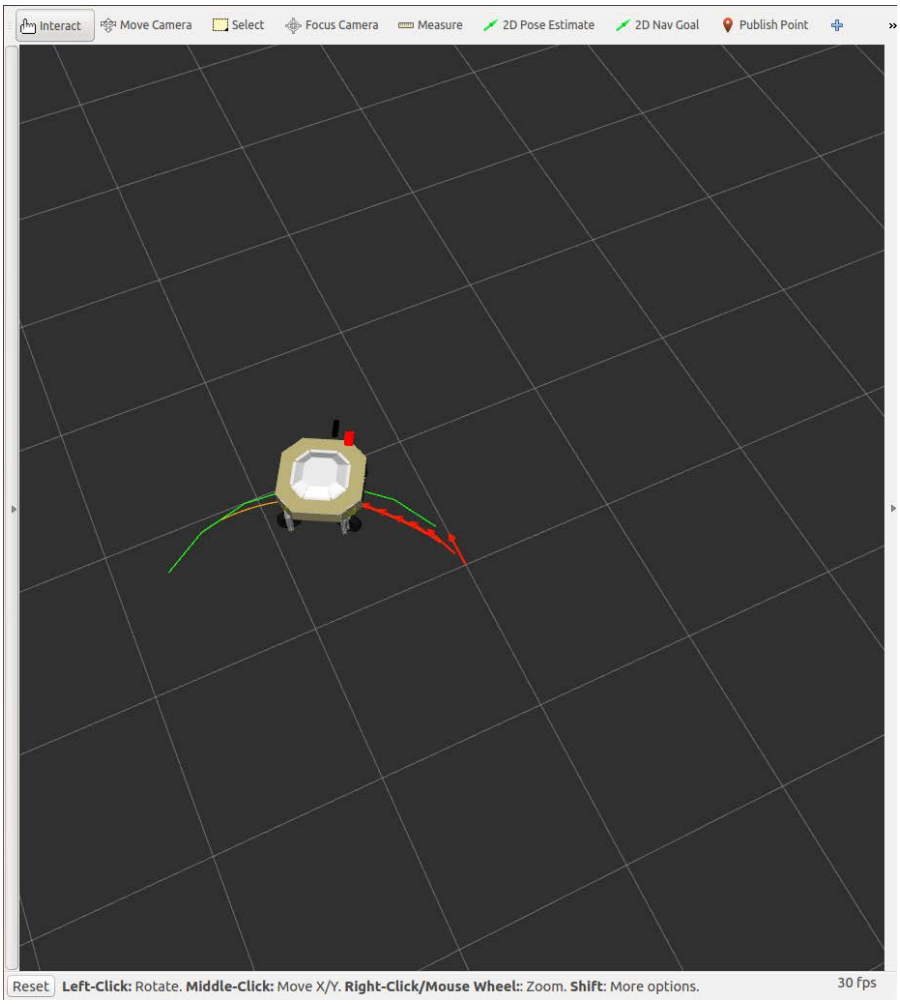


Figure 4.16: MARIO 2D navigation visualization in *Rviz*, green path: global path generated by the trajectory planner, orange: local planer correction path; red path: the travelled odometry from the robot state estimation system.

robotic software in a simulation environment before implementation on the real system.

Chapter 5

Fuzzy Singularity Avoidance for Optimal Kinematic Control

5.1 Introduction

In the previous chapter, the kinematic modelling and model-based controller (MBC) of MARIO were described. As stated in Section 4.3.1, the MBC performance decreases in the presence of kinematic singularities. For this aim, the singularity problem for this class of robots is discussed and formulated. Mathematical singularities are solved by presenting the idea of switching between different steering scenarios, while kinematic singularities still occur in one form of steering, which places the ICR on one of the steering axes. This kinematic singularity can result in damage to the actuator and the steering axis, and should be avoided. A novel fuzzy logic based approach is used to avoid a singular region around each steering axis, so that it controls the ICR by avoiding the singular region. Simulation results shows the effectiveness of the methodology in the kinematic singularity avoidance. Experimental results on MARIO validate the usefulness of the proposed methodology and its applicability to any other non-holonomic omnidirectional platform with three or more active steering and driving wheels.

5.2 Methodology

This section provides the formulation of the kinematic model and the mathematical and kinematic singularity of MARIO. The fuzzy singularity treatment including a brief presentation of the fuzzy inference system and fuzzy modelling is described.

5.2.1 Kinematic Model

The kinematic diagram of MARIO is shown in Figure 5.1. The robot frame is assumed to be a rigid body with planar motion. The formulation to calculate the linear and angular velocities, and the slip angle for each wheel are provided in the Equations 4.3- 4.6. The kinematic model formulated as above does not represent the kinematic singularity properly. For this reason, the ICR representation is as follow:

$$ICR_{[x,y]} = [-dir_y, dir_x] \frac{\|v_c\|}{|\dot{\theta}|}, \qquad dir_{[x,y]} = \frac{v_c}{\|v_c\|}, \tag{5.1}$$

where in Equation 5.1, $ICR_{[x,y]}$ is the Cartesian coordinates of ICR in the robot frame. Robot linear velocity unit vector is defined as $dir_{[x,y]}$. The polar representation of ICR is as follow:

$$icr = \|ICR\|, \qquad \lambda = arctan2\,(ICR_y, ICR_x), \tag{5.2}$$

where in Equation 5.2, icr is the magnitude, and λ is the angle of the ICR vector with respect to the x axis of the platform for representation in polar coordinates. Both linear velocity and steering angle of each wheel can be reformulated using ICR representation as:

$$v_i = \theta \left\| \overrightarrow{W} - \overrightarrow{ICR} \right\|, \qquad \psi_i = arctan\left(\frac{ICR_y - W_{iy}}{ICR_x - W_{ix}}\right) + \frac{\pi}{2}, \tag{5.3}$$

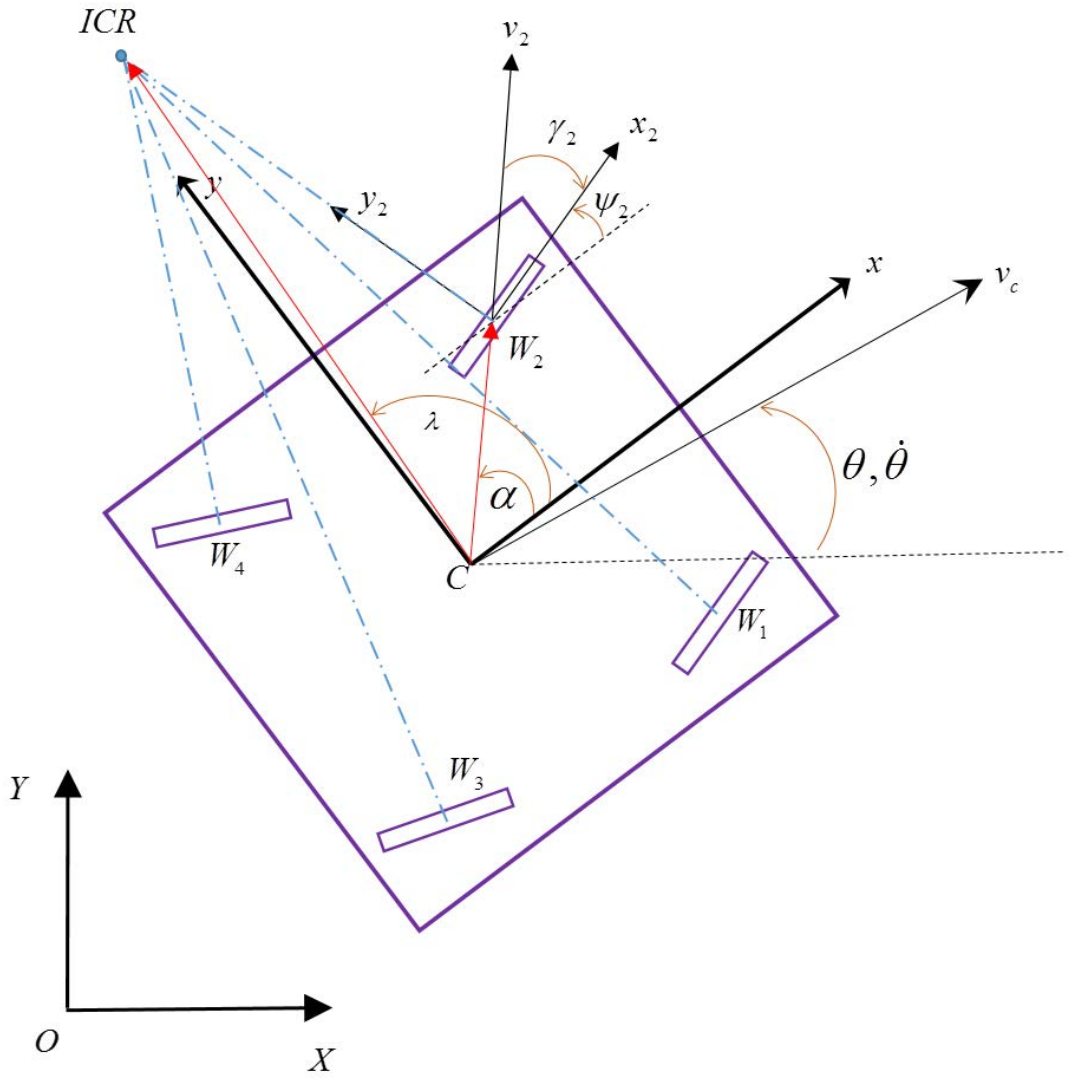


Figure 5.1: The kinematic diagram and the notations of a 4WD4S.

Mathematical singularities introduced by null linear or angular velocity are solved through switching the steering mode to one specific mode discussed in Section 4.3.1. Kinematic singularities happen with non-zero values of both linear and angular velocities, resulting in ICR located or passing through one of the steering axes. In this situation, an infinite number of solutions are possible to define the steering angle for this wheel and the joint steering rate increases unboundedly. The preferred solution is to guide and push the ICR out of the singular region optimally, so it avoids the problem.

5.2.2 Singularity Treatment

The proposed solution is to define a circular singular region around each wheel axis with the radius size of q centred from the wheel axis. If the ICR is located in or passing through this region, the aim is to direct it to the edge of the region to avoid the singularity. The singular region representation is shown in Figure 5.2. By calculating δ_1 and δ_2, the corrected ICR ($\overrightarrow{ICR}$) can be directed to the lower or upper edge of the singular region in the same direction of the old ICR. The solution can be achieved geometrically, but it suffers itself from singularity for the case when the $\overrightarrow{ICR}$ vector is aligned with the $\overrightarrow{W}$ vector ($\rho = 0$), which produces no solution.

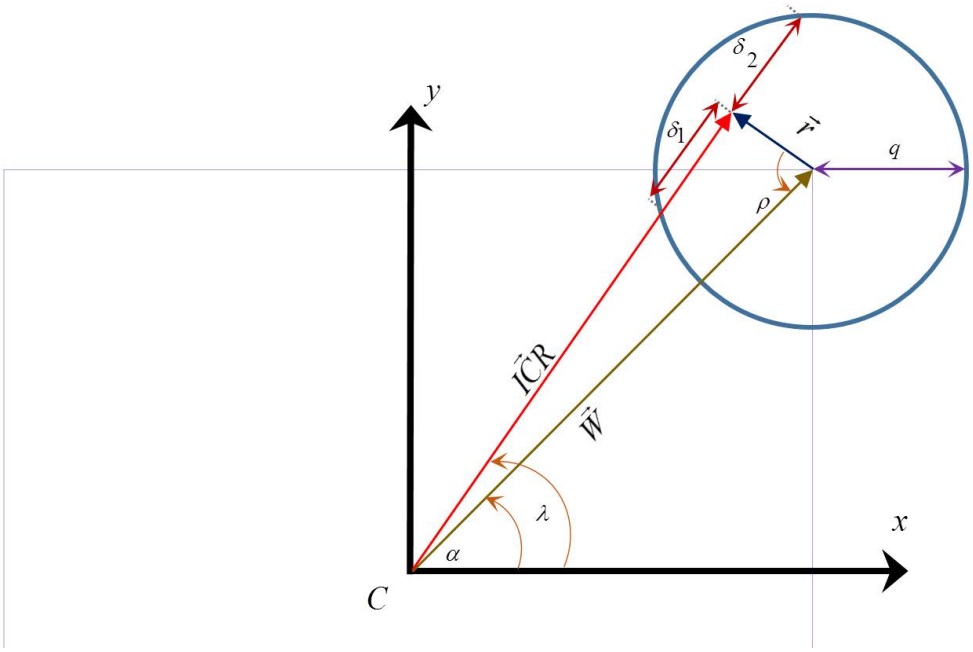


Figure 5.2: Singular region representation and the notations.

To avoid the mathematical singularity in the geometrical approach of controlling ICR, a simple and efficient solution is to use a fuzzy inference system to control the position of the ICR based on the specific inputs.

5.2.3 Fuzzy Singularity Avoidance

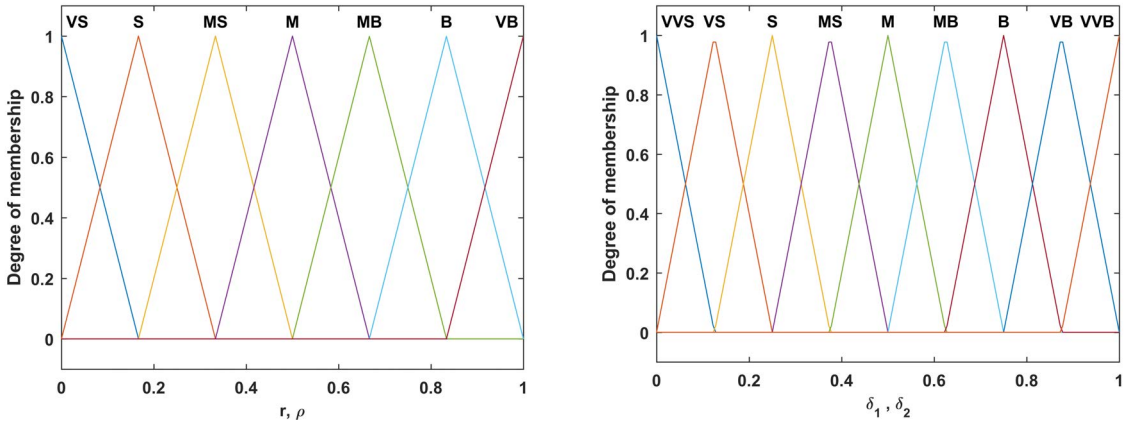
In Figure 5.2, it can be seen that for any ICR in the defined singular region, parameters $\overrightarrow{r}$ and ρ are known and can be calculated as:

$$\overrightarrow{r} = \overrightarrow{W} - \overrightarrow{ICR}, \qquad \rho = \angle\left(\overrightarrow{r}, \overrightarrow{W}\right) \tag{5.4}$$

The variations in the magnitude of vector $\overrightarrow{r}$ and angle ρ directly affect the size of δ_1 and δ_2. As discussed earlier, a geometrical approach can be used to calculate δ_1 and δ_2 , but the mathematical singularity happens when $\overrightarrow{ICR}$ is aligned to $\overrightarrow{W}$, where $\rho = 0$. The mathematical singularity can be solved by fuzzy modelling of a system to estimate δ_1 and δ_2 based on the two inputs $\overrightarrow{r}$ and ρ . This fuzzy system acts as a fuzzy controller that controls the output δ_1 and δ_2 based on the two inputs, the magnitude of $\overrightarrow{r}$ vector and ρ angle. The followings are the steps to model the fuzzy control system.

5.2.3.1 Fuzzification

The first step in the design of the fuzzy controller is the fuzzification, which converts the values of the actual inputs, r (magnitude of the vector $\overrightarrow{r}$), and ρ, into the linguistic values provided in Table 5.1. This conversion is obtained using fuzzy membership functions (MF) of the fuzzy set for each input. The inputs are normalized in the range of $[0\,1]$. The membership degree of an element in a fuzzy set is defined by the value of the MF. The value 0 represents the non-membership of the element to the fuzzy set, while 1 means full membership. Figure 5.3 shows the fuzzy sets including the MFs and the linguistic terms for fuzzification of actual inputs r and ρ (Figure 5.3a), and the outputs $\delta_1$and δ_2 (Figure 5.3b). MFs in a fuzzy set can have different shapes, here the triangular MFs have been used due to the simplicity of the definition and interpretation for both inputs and outputs.



(a) inputs r and ρ, (b) output δ_1 and δ_2,

Figure 5.3: Input and output fuzzy sets and MFs.

5.2.3.2 Fuzzy Rule Base

The second step of modelling the fuzzy control system is the setup of the fuzzy rules. The rules are created based on the available knowledge of the effect of the input variables on the output variable. For this fuzzy system, a total number of 49 fuzzy rules are needed to be defined as each of the inputs has 7 MFs in its fuzzy set. The fuzzy rules are given in Table 5.2. As an example, it is read as follows:

$$\textit{IF } (\ r \textit{ is } VS\ ) \textit{ AND } (\rho \textit{ is } VB\ ) \textit{ THEN } (\delta_1 \textit{ is } M\ ).$$

The fuzzy rules are created based on the visual and geometrical representation of the problem created in SolidWorks.

The fuzzy inference system (FIS) defines the output result of a rule. Common methods that have been used here are max-min method, where minimum of two fuzzified values is mapped to output as the AND-method and maximum as OR-method. Figure 5.4 shows the

Table 5.1: Linguistic input and output variables and associated quantification.

Inputs (r, ρ)	*Outputs* (δ_1, δ_2)
	***VVS**: Very Very Small = 0.0*
***VS**: Very Small = 0.0*	***VS**: Very Small = 0.125*
***S**: Small = 0.1667*	***S**: Small = 0.25*
***MS**: Medium Small = 0.3333*	***MS**: Medium Small 0.375*
***M**: Medium = 0.5*	***M**: Medium = 0.5*
***MB**: Medium Big = 0.667*	***MB**: Medium Big = 0.625*
***B**: Big = 0.8333*	***B**: Big = 0.75*
***VB**: Very Big = 1.0*	***VB**: Very Big = 0.875*
	***VVB**: Very Very Big = 1.0*

Table 5.2: Fuzzy rules of the fuzzy control system.

$r\backslash\rho$	***VS***	***S***	***MS***	***M***	***MB***	***B***	***VB***
VS	*M*	*M*	*M*	*M*	*M*	*M*	*M*
S	*MS*	*MS*	*M*	*M*	*M*	*MB*	*MB*
MS	*MS*	*MS*	*MS*	*M*	*MB*	*MB*	*MB*
M	*S*	*S*	*MS*	*M*	*MB*	*B*	*B*
MB	*VS*	*S*	*S*	*M*	*MB*	*B*	*VB*
B	*VS*	*VS*	*S*	*MS*	*B*	*VB*	*VB*
VB	*VVS*	*VVS*	*VVS*	*MS*	*B*	*VB*	*VVB*

plotted output variable δ_1 against the two input variables. As an example, $r = 1$ and$\rho = 1$ results $\delta_1 = 1$.

5.2.3.3 Defuzzification

The last step of the fuzzy control system is Defuzzification of the fuzzy values to a crisp output value as the output of the fuzzy control system. Many different methods are used to defuzzify the fuzzy values. The Centroid (centre of the gravity) is a common useful technique that is

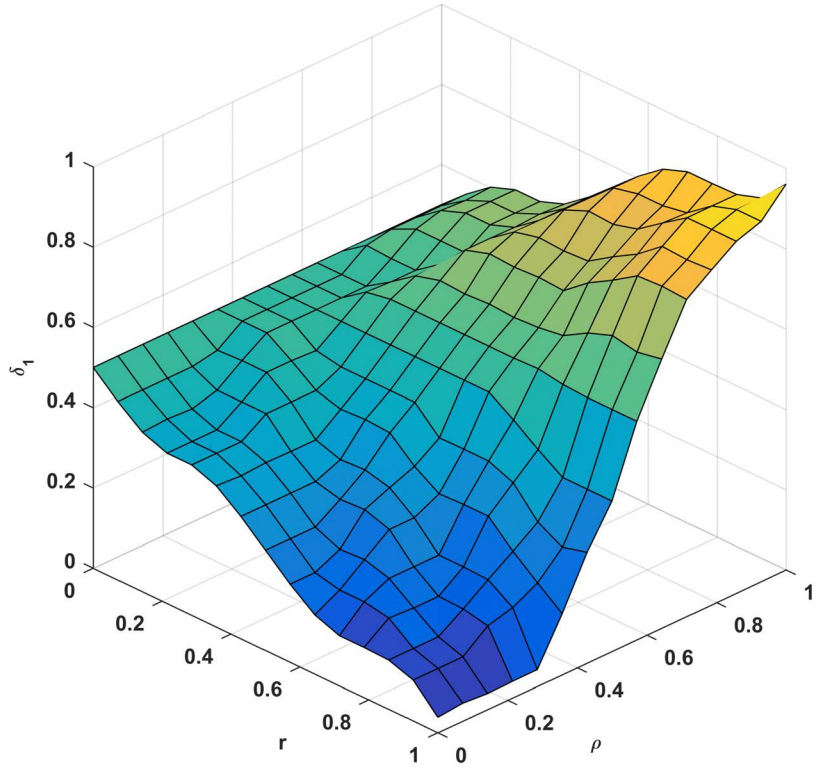


Figure 5.4: Fuzzy input-output surface.

used for this fuzzy control system which returns the centre of area under the aggregated curve mapped on the output MFs. The crisp output δ_1 using the centroid is generated by:

$$\delta_1 = \frac{\sum_{i=1}^{49} \bar{\delta}_{1i}\mu_i\left(\bar{\delta}_{1i}\right)}{\sum_{i=1}^{49} \bar{\delta}_{1i}}, \tag{5.5}$$

The input and output variables can be normalized for generalization and application of the designed system on any other model. The range for input r is $[0\ q]$(where q is the radius of the singular region), and the range for the input ρ is $[0\ \pi]$. The range for the outputs δ_1 and δ_2 is $[0\ 2q]$. The same fuzzy control system design can be used to estimate δ_2, as the inputs to the system are r and $\pi - \rho$. Symmetrical characteristic of the singular region allows to use the

same fuzzy control system design for the second half of the singular region. The fuzzy control system structure is shown in Figure 5.5.

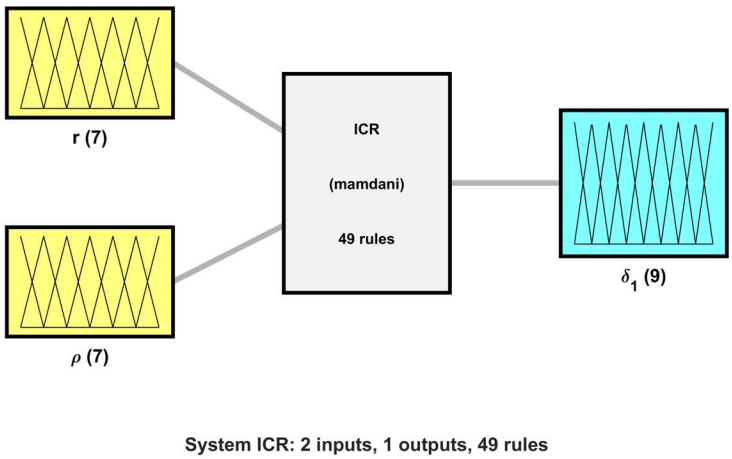


Figure 5.5: Fuzzy control system model for singularity treatment.

After the estimation of δ_1 and δ_2 , their values are used to correct the *ICR*. A simple logical expression is used to decide whether *ICR* should be added to δ_2 or subtracted from δ_1 to keep the ICR out of the singular region. In any case, the decision is made based on the previous state if δ_1 or δ_2 had been used. The decision formulation is as follows:

$$\overline{ICR_i} \begin{cases} ICR_i - \delta_{1i} & if \quad \delta_{1i-1} < \delta_{2i-1} \\ ICR_i + \delta_{2i} & if \quad \delta_{1i-1} > \delta_{2i-1} \end{cases} \tag{5.6}$$

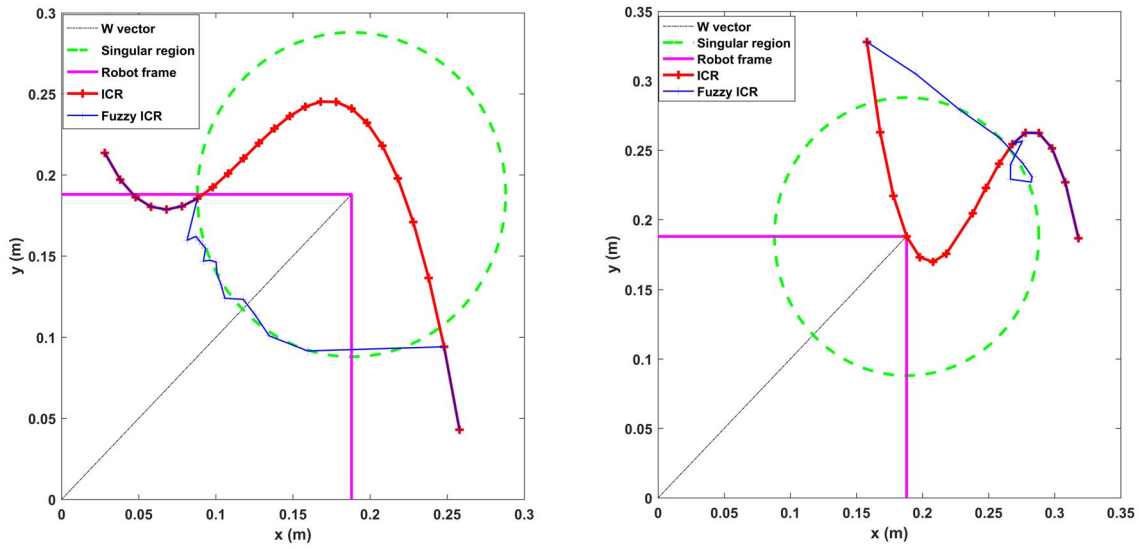
where in Equation 5.6, $\overline{ICR_i}$ is the modified *ICR* for the current step, $\delta_{1i-1}$and δ_{2i-1} are the values of the previous $\delta_{1i}$and δ_{2i}. The wheel speed and steering rates are also updated based on the $\overline{ICR_i}$ using the formulation stated in Equation 5.3.

5.3 Experimental Results and Discussion

This section provides the simulation results of the proposed methodology and the experimental validation on MARIO. The simulation results were obtained by simulating the ICR path to pass through the singular region around a wheel of MARIO. Based on the proposed fuzzy method, the expected result should cause the ICR path to avoid the singular region and pass around the singular region boundary.

Figure 5.6 presents the simulation results in two different scenarios. In Figure 5.6a, the original ICR (red path) enters the singular region and it is pushed down on the lower edge of the singular region, while it is pushed up to the higher edge of singular region in Figure 5.6b. The new corrected ICR (blue path) is calculated by the fuzzy system, while the direction of avoidance is decided by the conditional statement provided in Equation 5.6. As the results show in both cases, the ICR is not directed exactly on the border of the singular region. This is due to the fuzzified/defuzzified result of the fuzzy system and the quantification of the linguistic variables used in MFs.

Due to the change in the ICR profile in the singularity avoidance process, the velocity information should also be updated. The updated ICR determines the steering rate for each wheel while the driving rate (wheel velocity) needs to be updated based on the new corrected ICR as stated in Equation 5.3. To evaluate the velocity updates and profiles during the singularity avoidance, experiments are conducted on the MARIO platform. The experiment is carried out by causing the input velocity to place the ICR on one wheel axis and analyse the system response with and without the fuzzy singularity avoidance system. The results are provided in Figure 5.7. The input velocity to the robot base frame ($v_x = 0.1^{\mathrm{m}}/_{\mathrm{s}}$, $v_y = -0.1^{\mathrm{m}}/_{\mathrm{s}}$, $\dot{\theta} = 0.53^{\mathrm{rad}}/_{\mathrm{s}}$) places the ICR on the second wheel axis at $ICR_{[x,y]} = [0.188, 0.188]$.

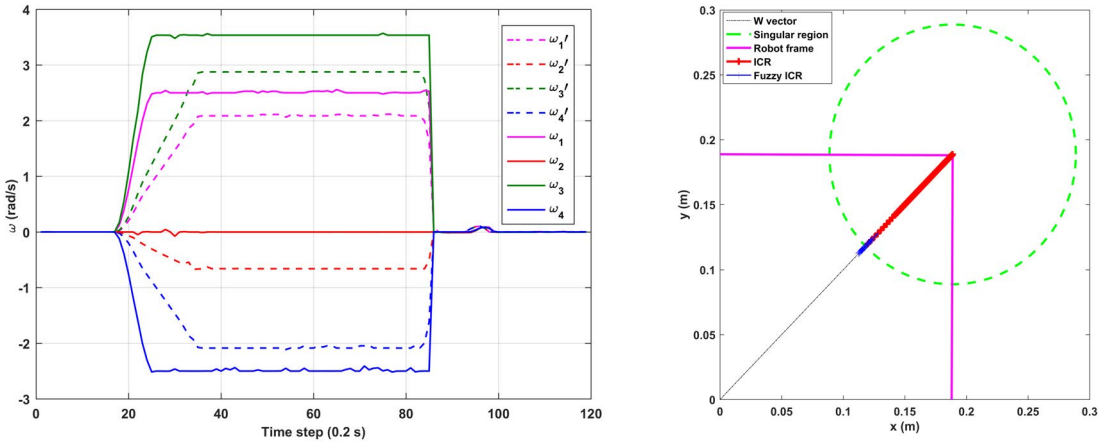


(a) (b)

Figure 5.6: Simulated ICR results passing the singular region in two scenarios.

The results presented in Figure 5.7a show the driving rate of each wheel in both scenarios with (dashed lines) and without (solid lines) the singularity avoidance. The ICR was placed on the second wheel's axis, so the second wheel is affected by the singularity and the kinematic controller produces zero velocity for the affected wheel (ω_2). All the other three wheels rotate around the second wheel's axis, which can cause damage to the steering axis and the wheel itself. By avoiding the singularity using the fuzzy control system, the ICR is placed on the border of the singular region (Figure 5.7b). The wheel velocities are different because of the updated fuzzy ICR, in which the second wheel driving rate would not be zero ($\acute{\omega}_2$). Figure 5.7b shows the ICR path during the experiment and the fuzzy ICR as the result of the singularity avoidance system.

The proposed method can be applied to any non-holonomic omnidirectional wheeled mobile robot with at least three active steering/driving joints. The proposed method takes normalized inputs and output normalized values, so it is independent from the kinematic of the system and it can be customized for any other platform.



(a) velocity profile of each wheel with and without singularity avoidance,

(b) original and corrected ICR during the experiment,

Figure 5.7: Singularity avoidance results on MARIO.

5.4 Summary

In this chapter, singularity problem in the MARIO MBC was studied. In general, Nonholonomic omnidirectional wheeled mobile robots suffer from both mathematical and kinematic singularities. Kinematic singularity happens when ICR places or passes through the steering axes, which causes singularity in defining the steering rate, while the steering rate of the singular joint increases unboundedly, gets over the mechanical and electrical limits of the actuator and can damage it. In this chapter, it is aimed to provide a solution in form of a fuzzy control system for the ICR to avoid a singular region around the steering axis, so that the kinematic singularity is avoided. The fuzzy control system controls and directs the ICR on the border of the defined singular region while it is passing through the singular region. The simulation and experimental results proves the practicality and usefulness of the proposed method in handling the kinematic singularity on MARIO.

Chapter 6

Enhanced Stereo Visual Odometry Using Sensor Fusion for Non-GPS Localization

6.1 Introduction

This chapter and the next two chapters are concerned with the second major contribution of this thesis on achieving an accurate and reliable localization for navigation of MARIO in GPS-denied environments such as orchards. This chapter presents investigations through an experimental study and analysis on utilizing stereo visual odometry (SVO) for vision based localization of MARIO in outdoor GPS-denied environments. As reviewed in Chapter 2, vision based localization systems are suffering from the accumulative drift over time. To overcome this drift, different methods were addressed including sensor fusion and computational optimization methods.

In this chapter, two of the known open source SVO libraries are utilized for analysis and comparisons. The performance of both SVO systems are analysed through the experimental study. The SVO drift is analysed in form of rotational and translational. Sensor fusion using *EKF* is employed to improve the rotational drift by fusing the SVO translational data and the IMU rotational data. The effect of sensor fusion is analysed in decreasing the SVO drift by

improving the rotational drift. Sensor fusion improves the results by reducing the rotational error and consequently minimizing the overall drift of SVO. The final remarks suggest required future work presented in Chapter 8 to improve the translational drift using a non-sensor fusion method.

6.2 System Setup

Typically a VO system includes one or more synchronized cameras, image processing algorithms and the processing unit. In a SVO system, at least two cameras are needed. This section presents the system setup and SVO methods that have been used to achieve the vision based motion estimation.

6.2.1 VO Algorithms

VO algorithms focus on the estimation of ego-motion of an agent frame-to-frame from the input camera images. These algorithms estimate the pose of the agent incrementally by processing and evaluating changes in image frames. In a monocular VO system, the relative 3D motion estimation is computed from a 2D image data. Therefore, all the measurements need to be scaled by an unknown factor to convert to a metric scale. The scale factor in monocular VO system adds a degree of uncertainty in the estimation. In a calibrated stereo camera system, scaled measurements can be obtained by triangulation of 3D coordinates of matched points in a single left/right image pair. Due to the importance of estimation accuracy and avoiding the uncertainty in the measurements, SVO has been utilised for MARIO.

Two open source algorithms have been used in this experiments, *fovis* [48] and *libviso2* [43]. Both of these algorithms provide 6 DOFs odometry.*Fovis* has been used successfully for micro aerial vehicles and *libviso2* has been tested on cars. Both algorithms use feature based techniques to estimate the motion. The basic steps for a SVO system is as follows:

1. Capturing the left and right stereo images at one frame;
2. Features detection on the captured images;
3. Feature matching/tracking;
4. Inlier detection on the matched features to reject the outliers;
5. Triangulation to achieve 3D coordinates of the matched features using the camera calibration parameters;
6. Motion detection from the 3D coordinates in two consecutive frames.

A more detailed description of SVO is provided in Chapter 7, but a brief description of *fovis* and *libviso2 is as follows.*

6.2.1.1 *Fovis*

Fovis uses *FAST* feature detector to extract features from the left captured image. Right image is used for disparity map to get the depth information for each pixel. Features with no depth information are discarded. Feature matching is achieved using *SIMD* instructions which scores the match between two features. Inlier detection is performed using a greedy algorithm to approximate the maximal clique in a graph of consistent matched features to remove the bad matches. Finally, motion is estimated through minimization of the 3D reprojection error using Gauss-Newton method.

6.2.1.2 *Libviso2*

Libviso2 algorithm estimates the 6 DOFs motion of the moving monocular or stereo camera. In a stereo system, motion estimation is performed by matching the extracted robust sparse features. Depth information is achieved by triangulation and minimization of the reprojection error of sparse matched features to compute the 3D coordinates. The input to the algorithm is

a set of matched features between four images, the left and right images of two consecutive image frames. Feature detection is performed by blob and corner detectors. To remove the outliers, *RANSAC* is applied to increase the robustness of motion estimator [54].

6.2.1.3 VO algorithm performance

The performance of both *fovis* and *libviso2* VO algorithms depend on factors such as algorithm parameters for each stage and hardware performance. A comparison of both algorithms performance has been performed by [37] that shows *fovis* has a better runtime than *libviso2*. Table 6.1 shows the comparison of the two algorithms on the same test dataset and hardware specifications.

Table 6.1: Computational Performance of *libviso2* vs. *fovis*.

Method	***Features***	***Mean Runtime (ms)***	***Average CPU Usage (%)***
Libviso2	2D Visual Features	39.5	29.8
Fovis	2D Visual Features and Depth	20.3	13.5

6.2.2 EKF Algorithm

The EKF algorithm used here is from the ROS *robot_localization* package [72]. This package supports input data from different odometry sources providing *nav_msgs/Odometry* type ROS message. Furthermore, it also supports input data from IMU in form of *sensor_msgs/Imu* ROS message. EKF Formulation explained in [101] can be described as a non-linear dynamic system with the robot state of:

$$\hat{x}_{k|k-1} = f\left(\hat{x}_{k-1|k-1}\right) + G_{k-1|k-1}, \qquad y_{k|k-1} = h(\hat{x}_{k|k-1}) + U_{k|k-1}. \tag{6.1}$$

where in Equation 6.1, $\hat{x}$ is the robot 3D pose state at time k, G is the normally distributed process noise, and f is the non-linear state transition function employed as a standard 3D kin-

ematic model. The measurement at time k is defined as y, h represents the non-linear sensor model which transfers the pose state into the measurement space, and U is the measurement noise with a normal distribution. The algorithm functions in two stages of prediction and update. In the prediction stage, the current pose state and error covariance are projected forward in time defined as:

$$\hat{x}_{k|k-1} = f\left(\hat{x}_{k-1|k-1}\right), \qquad P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k. \tag{6.2}$$

where in Equation 6.2, P is the estimate error covariance which is projected through F , the Jacobian of f, and further added by Q which is the process noise covariance. The update stage is then performed to achieve the pose estimate through below formulation:

$$K_k = P_{k|k-1} H_k^T \left(H_k^T P_{k|k-1} H_k^T + R_k\right)^{-1}, \tag{6.3}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(y_k - H_k \hat{x}_{k|k-1}), \tag{6.4}$$

$$P_{k|k} = (I - K_k H_k) P_{k|k-1} (I - K_k H_k)^T + K_k R_k K_k^T. \tag{6.5}$$

where in Equations 6.3, 6.4, and 6.5, A Kalman gain K is calculated using the observation matrix H, measurement covariance R , and P. This gain is used to update the state vector and the covariance matrix P. Figure 6.1 shows the Kalman Filter both stages to achieve state estimation at time k.

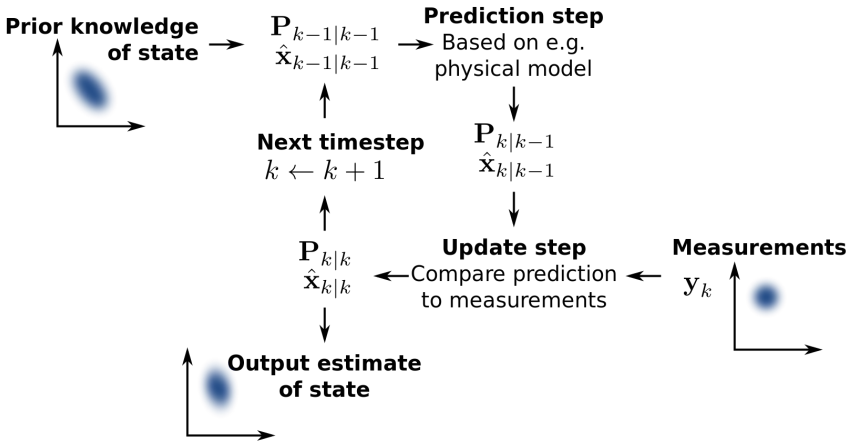


Figure 6.1: The Kalman Filter algorithm, (Obtained from wikipedia.org/wiki/Kalman_filter).

6.2.3 The Hardware Setup

ZED stereo camera developed by Stereolabs is a lightweight depth camera based on the passive stereo vision. It outputs high resolution video that contains two synchronized left and right image streams, and generates real-time depth and disparity maps of the environment using a graphical processing unit (GPU). In stereo video mode, it outputs ultra-high definition 2.2k video with a frame rate of 15 fps while the highest frame rate (100 fps) can be achieved in VGA video mode. Depth mode generates the depth map in the range of 1 to 15 meters with the same resolution as video mode. Frame rate depends on the speed of GPU that runs the ZED software development kit (v 0.9, Stereolabs) to create the depth map. Highly optimized algorithms and auto-calibration system are significant features of the ZED camera. Camera calibration is one of the major issues for stereo vision systems especially in real time. Figure 6.2 shows the ZED stereo camera.

Nvidia Jetson Tegra K1 (Jetson Tk1) board has been used to run ZED SDK. The Jetson TK1 board has a quad-core ARM CortexA15 processor and a Keppler GPU with 192 CUDA cores. It runs a pre-installed Linux4Tegra OS which is based on Ubuntu 14.04. The ZED SDK requires OpenCV (v 3.2 Open Source Computer Vision Library) and CUDA 6.5 to function.

Figure 6.2: ZED stereo camera from Stereolabs.

The system also runs ROS indigo version to launch the camera using its ROS driver.

6.3 Experimental Results and Discussions

To evaluate the performance of each VO method, an experiment was conducted in an open field (Ilam Field, Christchurch, New Zealand). Navigating the robot in this open field assures good satellite coverage for the RTK-GPS unit to measure pose information as ground truth. The robot was driven on a $25m \times 15m$ commanded rectangle path for the first run and a $25m \times 25m$ path for the second run. All the sensors data were recorded for post processing using the rosbag tool from ROS during the test to enable supplementary experiments and evaluations using two different VO methods. Figure 6.3 shows the test field and the commanded path of the first run with more details shown in Table 6.2.

The recorded data including the stereo video was used as input for *libviso2* and *fovis* SVO algorithms. Figure 6.4 shows a captured scene from the left camera and the detected features by *fovis* in two consecutive frames.

Figure 6.3: Test location aerial image and navigation path by Google Earth Pro.

Table 6.2: Experiment details.

Item	*Run 1*	Run 2
Total trajectory length	80 m	100*m*
Average speed	0.25 m/s	0.25 m/s
Slope	0°	0°

To run and compare the outputs of two algorithms with ground truth in one global frame, the rosbag data was played back and the data from stereo camera including left and right images and camera info were fed to *fovis* and *libviso2* ROS packages simultaneously in one run. The results are shown in Figure 6.5. The blue graph is the travelled trajectory and estimated motion by the RTK-GPS as the ground truth. The red and purple graphs present the SVO estimation by *libviso2* and *fovis* respectively.

As the results in Figure 6.5 suggest, despite the fact that both visual odometers suffer from

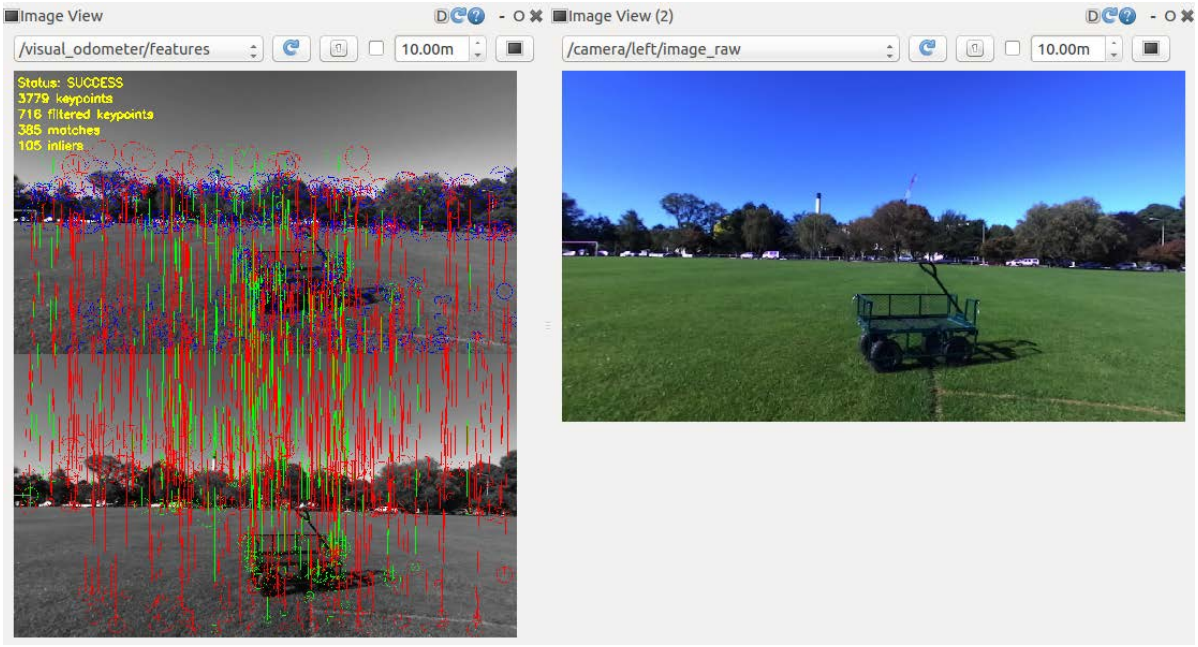


Figure 6.4: Features detected and matched by *fovis* and a captured scene from the left camera.

the drift, odometry by *libviso2* is much closer to the ground truth compared to *fovis* in the same run. *Libviso2* fails in the first turn and this causes a larger drift as it increases incrementally. *Fovis* in the second run completely fails the estimation. To quantify and compare the results numerically, average drift and error have been calculated. The numerical results are summarized in Table 6.3. Drift in form of reprojection error for each point is calculated and the average drift in form of Root-Mean Square Error (RMSE) is defined as:

$$RMSE = \sqrt{\frac{1}{n}\sum_{k=1}^{n}\left(\widehat{XYZ}_i - XYZ_i\right)^2}, \tag{6.6}$$

where in Equation 6.6, $\widehat{XYZ}$ is the reference data (e.g. RTK-GPS 3D coordinates) and $XYZ$is the measured data (e.g. SVO estimation) at frame or sample k.

Table 6.3 presents the quantified drift in form of translational drift and error, and the rotational drift. Translational error is achieved by the division of the translational drift by the total travelled path. Rotational drift is calculated based on the referenced IMU rotational data and the SVO rotational estimate in Euler angles. The average drift and error generated by *fovis* in

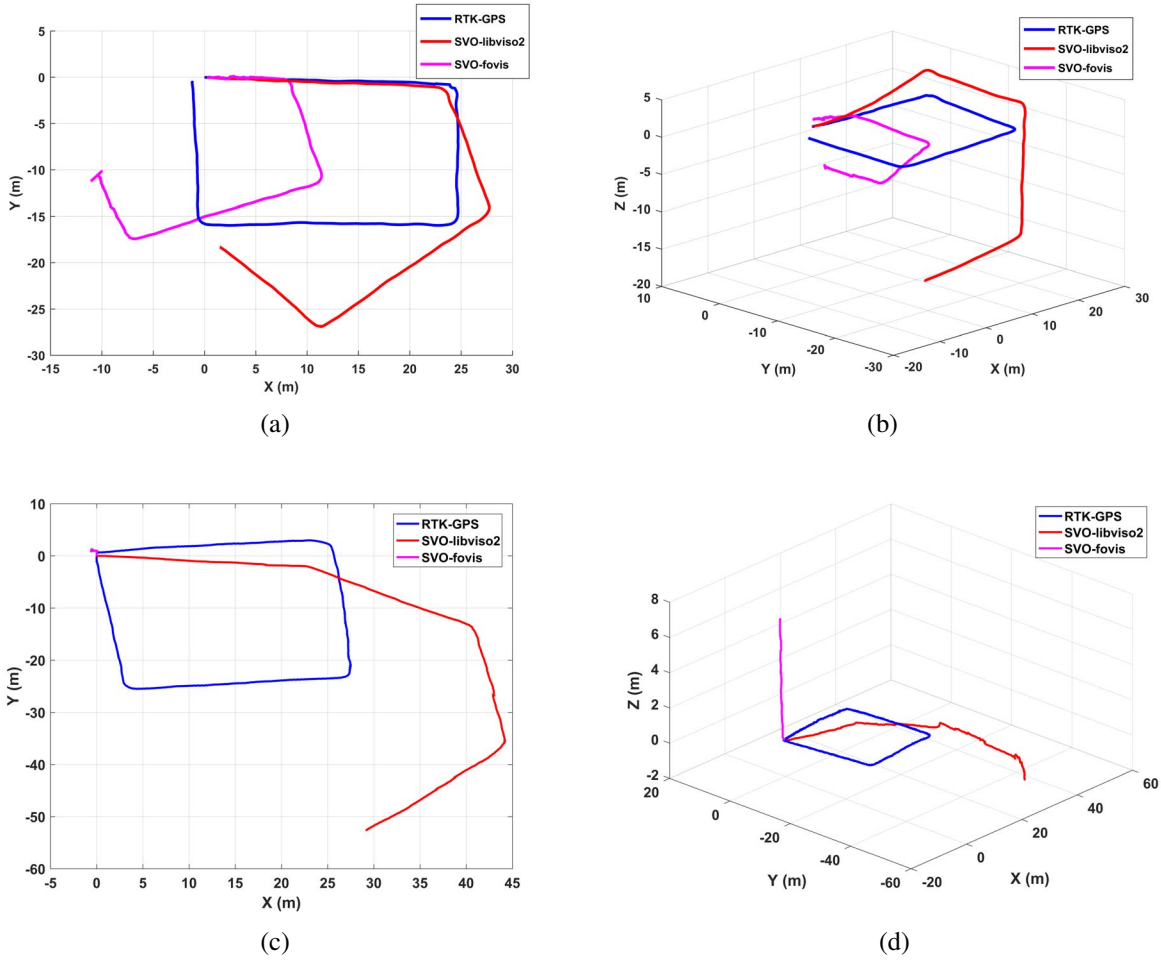


Figure 6.5: Results of *libviso2* and *fovis* estimations in comparison to the RTK-GPS as ground truth from the experiment in 2D and 3D plotted graphs: (a) 2D in run1; (b) 3D in run1; (c) 2D in run2; (d) 3D in run2.

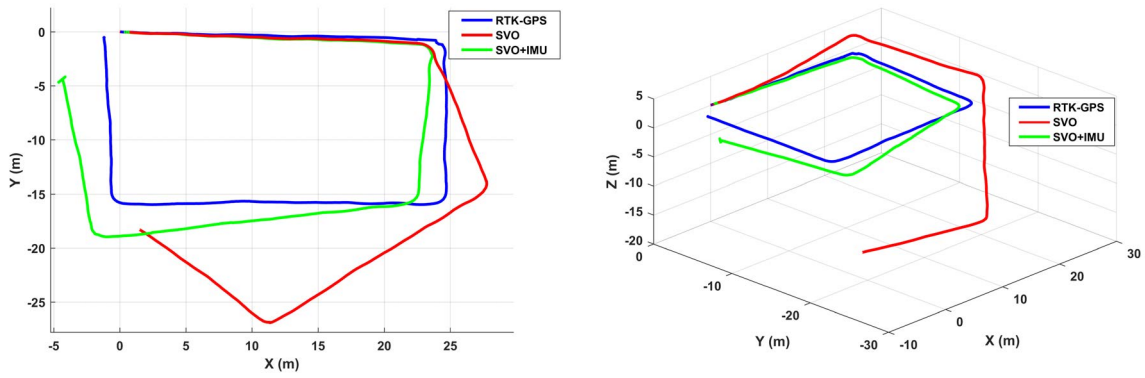
the first run is almost double of *libviso2*. This argument is not valid for the second run as *fovis* fails the experiment to estimate, while the drift and error quantification show better results.

The comparison of initial results shows the better performance of *libviso2* over *fovis* both in rotational and the translational drift for this experiments. From the results in Figure 6.5, *libviso2* fails to calculate the rotation in the first turn of the rectangle path and this causes the drift to remain high for the rest of the path. The numerical comparisons in this experiment quantifies the translational error while rotational error plays an important role in increasing

Table 6.3: Comparison results of drift and translational error for two methods.

Method	Translational Drift (m)	Translational Error %	Rotational Drift (rad)
SVO-fovis_run1	11.14	13.9	1.77
SVO-fovis_run2	13.09	13.09	0.67
SVO-libviso2_run1	6.48	8.1	0.69
SVO-libviso2_run2	21.11	21.11	0.62
SVO-libviso2_run1 + IMU	1.09	1.3	0.57

the drift. To further improve the results, *libviso2* was fused with IMU data to overcome the rotational error by the help of an extra sensor. To achieve this, *libviso2* (X_k , Y_k , Z_k) translational information and IMU rotational information ($roll_k^{imu}$, $pitch_k^{imu}$, yaw_k^{imu}) at time k are fused using *EKF* from ROS robot_localization package. The result is shown in Figure 6.6.



(a) (b)

Figure 6.6: The result from data fusion of *libviso2* and IMU using *EKF*.

As it can be seen in Figure 6.6, rotational drift has been reduced in first turn and as a result, SVO tracks the ground truth with lower drift. The average drift and translational error is shown in Table 6.3. Integration of *libviso2* and IMU has reduced the drift and translational error significantly. *EKF* has also improved the large elevation drift by *libviso2* in the whole run. The average drift value is 1.09 m and the translational error is 1.3% in a travel distance of

80 meters. To describe the effect of rotational error from SVO, Figure 6.7 shows the drift in 3 scenarios on the same input data. *"SVO-Libviso2"* and *"SVO-libviso2 + IMU"* have the same drift as the robot moves on a straight path toward the first turn. After first turn, it is seen that *libviso2* drift increases significantly compared to the integrated data. This causes continuous increase in the drift as the robot travels to the end of the path.

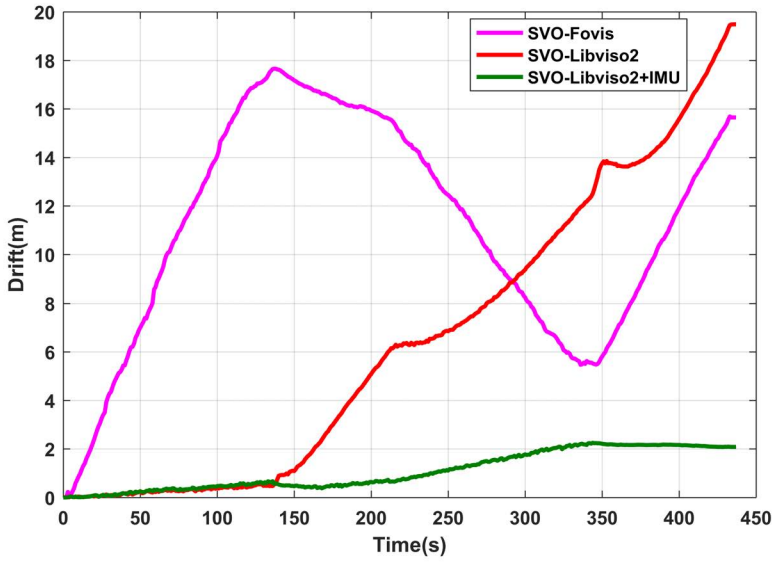


Figure 6.7: Translational drift between SVO estimations from fovis, *libviso2*, and *libviso2 + IMU*.

Sensor fusion using *EKF* for this experiment could overcome the rotational drift. Rotational drift correction resulted translational drift improvement in some extent, specifically by reducing the elevation drift through correcting the pitch angle of the SVO estimation. However, sensor fusion of SVO with IMU does not suppress or correct the drift completely, and the accumulation of the SVO drift after correction is still inevitable. The ideal for a non-GPS localization system is the lowest possible drift for an accurate pose estimation. Non-sensor fusion methods such as local optimisation methods reviewed in Section 2.4.2, have proven to reduce the SVO drift at each frame. The next chapter covers a non sensor fusion-method to

overcome the translational drift of SVO system.

6.4 Summary

This chapter presented the experimental study of utilising two of the open source SVO algorithms , *fovis* and *libviso2*, and the implementation of them on MARIO for localization in outdoor GPS-denied environments. Despite the main drawback of SVO in translational drift, it has been shown that the rotational error of SVO has a key role in increasing the translational drift in a larger extent. The average drift and translational error have been quantified, *fovis* had almost twice the error as *libviso2*. To overcome the rotational error, rotational information of IMU has been integrated with *libviso2* information using EKF. This improves the results as it reduces the rotational error, specially in the corners and consequently minimizes the overall drift by a factor of 6.23. Next chapter presents a more in-depth description of SVO and the SVO drift and provides a non-sensor fusion approach to decrease the translational drift of the selected SVO (*libviso2*) to improve the result of SVO for reliable localization in outdoor GPS-denied environments.

Chapter 7

Enhanced Stereo Visual Odometry Using Machine Learning

7.1 Introduction

This chapter presents a novel non-sensor fusion approach to improve the accuracy of the SVO by reducing the translational drift, aimed for a better localization of MARIO in GPS-denied environments such as orchards. For this aim, one of the known open source SVO libraries that was tested and selected for further development in Chapter 6, is considered as the core pose estimation system. To improve the translational drift, two machine learning approaches have been used. Radial Basis Function (RBF) Networks and Adaptive Neuro-Fuzzy Inference System (ANFIS) have been utilized to model a translational drift estimator. Both ANFIS and RBF network are trained using the training dataset, including the samples input and output parameters. The input parameters are the inlier percentage and the reprojection error of the SVO process at each frame, while the output is the translational drift ratio. Estimated translational drift is then used to correct the SVO drift at each frame. Both ANFIS and RBF network are validated and tested using the validation and test datasets. The experimental results are compared to present the improvements for the pose estimation of the SVO by estimation and

correction of the translational drift.

7.2 Stereo Visual Odometry

SVO is the process of estimating the ego-motion of an agent through extracting, matching, and tracking features frame by frame captured by a camera. In a monocular SVO system, the motion is estimated from the 2D image data; hence, the estimation should be scaled by a factor to convert the estimation into a metric measure. Such a problem for a SVO system using a calibrated stereo camera is solved by triangulation of 3D coordinates of matched features in a single stereo image (left and right) pair. Therefore, in this work a SVO system has been used.

The SVO from *libviso2* open source library has been employed [43] as it was tested and selected in the previous chapter based on its performance. *Libviso2* library provides algorithms for 6 DOFs motion estimation using monocular or stereo cameras. The general workflow of the SVO system of *libviso2* is as follows:

1. Features from the left and right rectified images are detected using blob and corner detectors;

2. 3D coordinates of matched features are computed through triangulation using the camera calibration parameters (intrinsic and extrinsic parameters);

3. Random Sample Consensus (*RANSAC*) is applied for the outlier rejection;

4. The estimation of motion is performed by the minimization of the reprojection error of the matched 3D features in two consecutive frames.

The triangulation to acquire 3D coordinates of feature point is expressed as:

$$d = P_u^l - P_u^r, \qquad P_z = \frac{bf}{d}, \qquad P_x = \frac{P_z\left(P_u^l - c_u\right)}{f}, \qquad P_y = \frac{P_z\left(P_v^l - c_v\right)}{f} \tag{7.1}$$

Where in Equation 7.1, d is the feature disparity, $P_{[x,y,z]}$ is the 3D coordinate of the feature point from the 2D image feature location in the left image $P^{l}_{[u,v]}$and right image $P^{r}_{[u,v]}$.The parameter f is the focal length of the camera in pixel and $c_{[u,v]}$ is the optical centre both in the unit of pixels and obtained from the camera calibration intrinsic matrix, while b is the camera baseline in meters and achieved from the camera calibration extrinsic matrix. The ZED stereo camera intrinsic and extrinsic parameters are based on the presented parameter in Figure 7.1 achieved through the calibration process is presented in Table 7.1.

Table 7.1: ZED stereo camera intrinsic and extrinsic parameters.

Parameter	Left camera	Right Camera	Stereo Camera
f_u	672.30	672.30	~
f_v	672.30	672.30	~
c_u	648.22	648.22	~
c_v	361.63	361.63	~
b	~	~	0.120

The stereo camera model and parameters are shown in Figure 7.1. The motion is estimated based on the feature sets from the two consecutive frames using minimization of the reprojection error and defined as:

$$T_k = g\left(P^{k-1}, P^{k}\right), \qquad T_k = \begin{bmatrix} R_k & t_k \\ 0 & 1 \end{bmatrix}, \qquad C_k = C_{k-1}T_k^{-1} \tag{7.2}$$

Where in Equation 7.2, T_k is the homogeneous 4×4 transformation matrix which represents the transformation of the camera from the previous frame to the current frame at time k, while C_k is the homogeneous 4×4 matrix representing the camera pose at time k achieved by matrix concatenation. Correspondingly, R_k represents the 3×3 camera rotation matrix of the current frame with respect to the previous frame, while t_k is the 3×1 translation vector defining the translation of the current camera frame from the previous frame. The function $g(.)$

refers to the Gauss-Newton iterative optimization to minimize the reprojection error which is a function of the extracted set of features P in the current frame k and previous frame $k-1$.

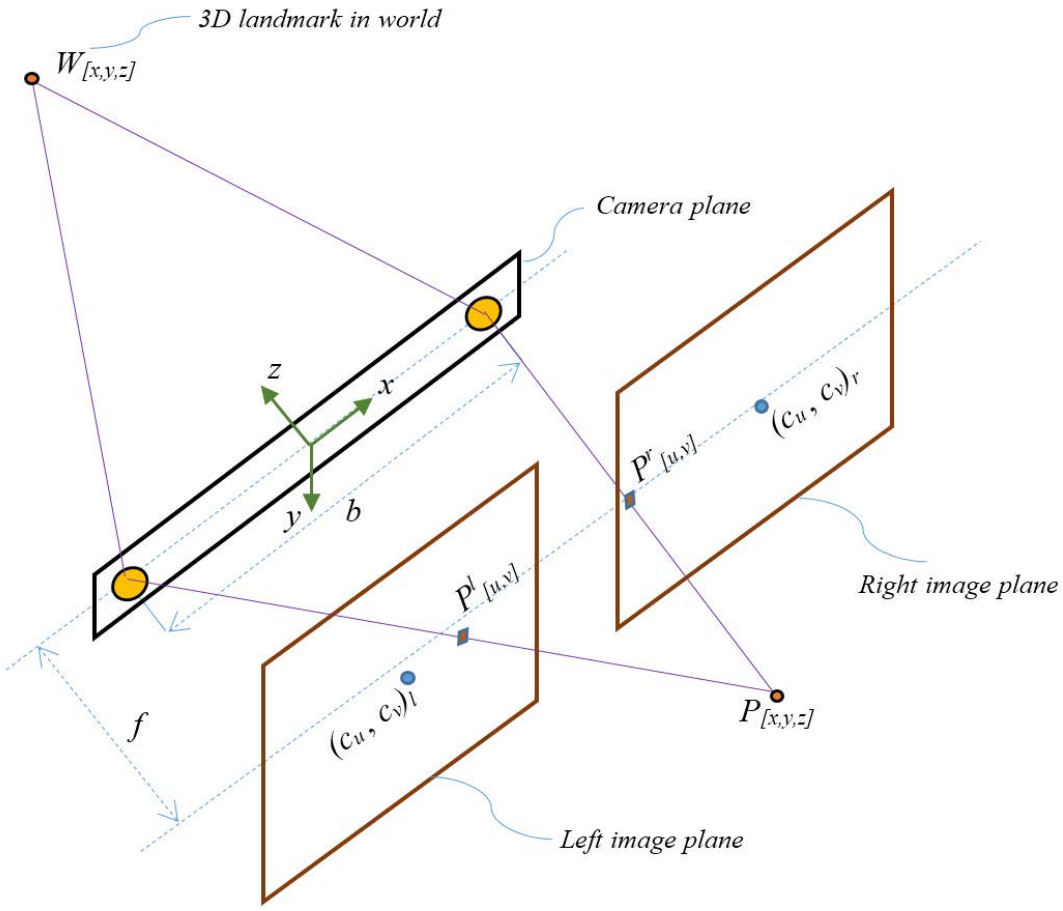


Figure 7.1: Stereo camera model parameters and landmark projection.

7.2.1 SVO drift nature

Current pose estimation of the camera in SVO is obtained by the concatenation of each transformation matrix at each time step. Each individual transformation has an uncertainty and the camera pose uncertainty propagates through the concatenation of all previous transformations. The source of uncertainty in transformation matrix has been a subject of interest for researchers while it has been defined mainly as the uncertainty or error in the feature matching process

[52, 64]. As discussed by Fraundorfer and Scaramuzza in [39] the SVO uncertainty propagation can be formulated as follows. Both camera pose C_k and transformation T_k matrices can be represented individually by a 1×6 vector as $\overrightarrow{C}_k$ and $\overrightarrow{T}_k$ including the 3D position in Cartesian coordinates (x, y, z) and 3D orientation in Euler angles (ϕ, θ, ψ). Each transformation $\overrightarrow{T}_k$ is represented by its mean and the 6×6 covariance matrix M_k. As stated in Equation 7.2, each camera pose C_k, which is a function of the current transformation T_k and the previous camera pose $C_{k-1}$with their covariances $M_k$and $\overline{M}_{k-1}$respectively. The combined covariance matrix for $\overrightarrow{C}_k$ $(\overline{M}_k)$ from the two M_k and $\overline{M}_{k-1}$, covariances is a 12×12 matrix and can be computed using the law of error propagation presented which applies a first-order Taylor approximation:

$$\overline{M} = J \begin{bmatrix} \overline{M}_{k-1} & 0 \\ 0 & M_k \end{bmatrix} J^T = J_{\overrightarrow{C}_{k-1}} \overline{M}_{k-1} J^T_{\overrightarrow{C}_{k-1}} + J_{\overrightarrow{T}_k} M_k J^T_{\overrightarrow{T}_k} \tag{7.3}$$

where in Equation 7.3, $J_{\overrightarrow{C}_{k-1}}$ and $J_{\overrightarrow{T}_k}$ are the Jacobians of the function $C_{k-1}\,T_k$ variables respectively. As it can be seen in Equation 7.3, the camera pose uncertainty is always increasing through transformation concatenation at each frame. Therefore, it is essential to reduce the uncertainties to as low as possible at each frame for each transformation to decrease the overall SVO drift. In this regard, our approach in this chapter focuses on the estimation and correction of SVO drift at time k to suppress the uncertainty for each individual estimated transformation.

7.3 SVO Drift Estimation and Correction

The SVO translational drift estimation is performed using a machine learning framework, where supervised training is employed based on a set of training data. The training dataset includes a large samples of input parameters representing some properties of the SVO process, and the quantified SVO drift as the output parameter. The trained and validated network can be used as a typical function approximation system by feeding new inputs to get the estimated

SVO drift as the output. For this aim, we use two of the machine learning approaches, RBF network and ANFIS, and evaluate their performance in solving this problem.

7.3.1 Drift estimation system parameters

The input parameters representing different characteristics of the SVO system are described as follows. In this context, 2 parameters are considered as the input parameters:

- The inlier percentage,
- The average error of the reprojected 2D feature points.

7.3.1.1 *Inlier percentage*

The matched feature points usually contain outliers, indicating incorrect data relationship. The outliers are usually caused by image noise, illumination variations or other disturbances which are not supported by the mathematical formulation of the feature detection system. Outlier rejection is required for an accurate motion estimation by the *SVO* system. *RANSAC* is a commonly used standard framework for outlier rejection. In this regard, different hypotheses of the motion model are computed from a set of sample feature points (3 samples in *libviso2*). Then these hypotheses are verified on the other feature points. The hypothesis that represents the maximum agreement, or in other words, a reprojection error less than a certain threshold (for *libviso2* 1.5 pixels) is selected as the model to estimate the motion, and its data (feature points) are considered as the inliers. The inlier percentage can be a good measure to quantify the performance of the feature matching process. The inlier percentage is achieved at each time from the number of inliers over the total number of features provided by the *RANSAC* process.

7.3.1.2 *Reprojection error*

As stated in Equation 7.2, a Gauss-Newton iterative optimization is used to extract the motion of the camera by minimizing the reprojection error. To evaluate the performance of the optimization process, the reprojection error is quantified to be used as an input parameter in the training process of the machine learning system. The reprojection error is formulated by simulating a new solution for the 3D feature coordinates of the current frame using the 3D feature coordinates of the previous frame and the transformation matrix in form of the average Euclidean distance, defined as:

$$\overline{P}^k_{[x,y,z]} = R^T_k \, P^{k-1}_{[x,y,z]} + R^T_k \, t_k, \qquad \overline{P}^k_u = f\left(\overline{P}^k_x / \overline{P}^k_z\right) + c_u, \qquad \overline{P}^k_v = f\left(\overline{P}^k_y / \overline{P}^k_z\right) + c_v, \tag{7.4}$$

$$\varepsilon_k = \frac{1}{n} \sum_{i=1}^{n} \left\| P^k_{[u,v]_i} - \overline{P}^k_{[u,v]_i} \right\|_2, \tag{7.5}$$

where in Equations 7.4 and 7.5, $\overline{P}^k_{[x,y,z]}$ is the 3D reprojected feature coordinates for the current frame k, $\overline{P}^k_{[u,v]}$ is the set of n reprojected 2D feature points for the current frame obtained from the 3D reprojected feature coordinates and the camera calibration intrinsic parameters, and ε is the reprojection error at time k.

7.3.1.3 *SVO drift ratio*

SVO drift ratio is considered as the output of the machine learning system since the aim is to estimate the SVO drift at each time k and correct the original SVO output to compensate the drift. In this research, the focus is to reduce the translational drift. Therefore, the translational drift ratio is quantified using the estimated motion by SVO and the GPS positioning data as

the ground truth at each time, k. The SVO drift ratio at each frame is formulated as:

$$E_k = \frac{\left\| GPS^k_{[x,y,z]} \right\|}{\left\| C^k_{[x,y,z]} \right\|}, \qquad E \in [0\ 1] \tag{7.6}$$

where in Equation 7.6, E is the computed SVO drift ratio at time k, $GPS_{[x,y,z]}$ is the 3D coordinates from the GPS, and $C_{[x,y,z]}$ is the estimated concatenated motion by VO. The drift ratio is expected to be in the interval of $[0\ 1]$. This is due to the nature of the SVO drift that is increasing over time. The computed drift ratio is used in the training process, while the estimated drift ratio as the output of the machine learning system on the test data is used to correct the original SVO data.

7.3.2 Machine learning framework

This section describes the machine learning framework that has been used for the estimation of the SVO drift ratio from the inputs, the inlier percentage and the reprojection error described in the earlier section. ANFIS and RBF Network are considered as the core of the machine learning framework. The structure of both systems are described as follows.

7.3.2.1 Adaptive Neuro-Fuzzy Inference System (ANFIS)

ANFIS functionally is a fuzzy inference system (FIS) in which the parameters of FIS have been obtained through a neuro-learning process using a training dataset rather than determination by an expert. First ANFIS model was proposed by Jang in [51] based on the Takagi-Sugeno-Kang (TSK) fuzzy model. This technique provides a multi-input-single-output system modelling in form of a fuzzy modelling procedure, while the membership function parameters are adjusted through a learning process.Figure 7.2 presents the general architecture of an ANFIS. ANFIS has five layers to build the fuzzy system including, I) fuzzy layer, II) product layer, III) normalized layer, IV) defuzzification layer, and V) output layer.

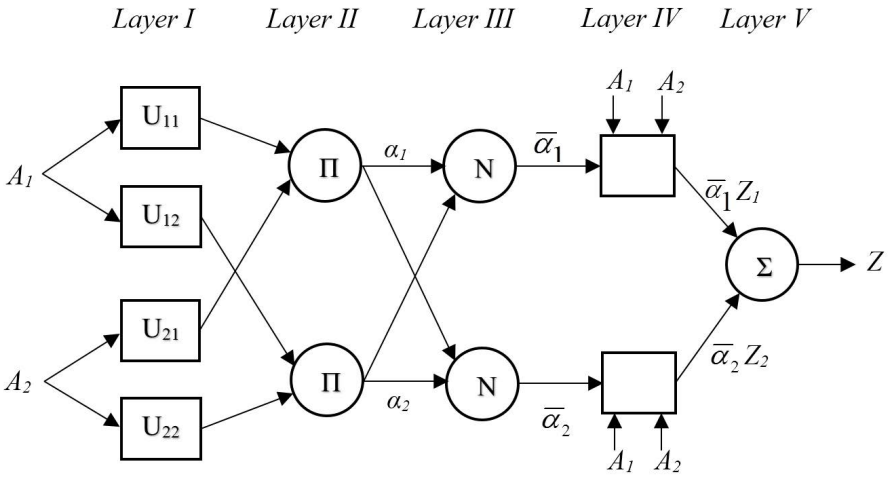


Figure 7.2: ANFIS architecture.

ANFIS training is supervised and the training dataset consists of inputs and their corresponding desired output values. For an ANFIS model with a first-order TSK FIS as depicted by Figure 7.2, consisting two inputs (A_1 and A_2) and one output (Z), the typical rule set can be presented as:

- Rule 1: if (A_1 is U_{11}) and (A_2 is U_{21}) then ($Z_1 = p_1A_1 + q_1A_2 + s_1$),
- Rule 2: if (A_1 is U_{12}) and (A_2 is U_{22}) then ($Z_2 = p_2A_1 + q_2A_2 + s_2$),

where U expresses the membership functions and p, q, and s are the assigned parameters during the training process. Each node in the first layer outputs membership graded values of inputs for the fuzzy sets. Different membership functions are available such as Gaussian, triangular, trapezoid, and sigmoid. A Gaussian membership function is defined as:

$$U_{ij} = \exp\left(-\frac{(c_{ij} - A_i)^2}{2\sigma_{ij}^2}\right), \qquad i = 1, \ldots, n, \tag{7.7}$$

where in Equation 7.7, U_{ij} is j_{th} member function of the i_{th} input, c_{ij} is the centre and σ_{ij} is the spread rate of the Gaussian membership function. These two parameters are defined as

the premise parameters. The second layer consists of fixed nodes $\prod$, which output the product of incoming inputs from the first layer. The output value represents α_j, the firing strength of each rule and for the k_{th} node is expressed as:

$$\alpha_j = \prod_{k=1}^{o} U_{ijk,} \tag{7.8}$$

The nodes in third layer labelled as N, output the average of the fuzzy rules weights obtained from the second layer:

$$\overline{\alpha}_j = \frac{\overline{\alpha}_j}{\sum_{j=1}^{n} \alpha_j}, \tag{7.9}$$

where in Equation 7.9, $\overline{w}_j$ is the normalized firing strength. The fourth layer multiplies the normalized firing strengths of all the rules with z which is a function of few parameters. The output of each node in this layer is given by:

$$\overline{\alpha}_j Z_j = \overline{\alpha}_j \left(p_j A_1 + q_j A_2 + s_j \right), \tag{7.10}$$

where in (10), p_i, q_i, and r_i were described earlier, are the consequent parameters set during the training process. The only node in the fifth layer sums up the incoming inputs and outputs the overall output value.

$$Z = \sum_{j=1}^{n} \overline{\alpha}_j Z_j, \tag{7.11}$$

Typically, ANFIS parameters are adjusted in the training process using the back-propagation (BP) or a hybrid learning algorithm that combines BP with Least-Square-Method (LSM). The optimal consequent parameters can be defined using LSM if premise parameters are considered as fixed values. While both premise and consequent parameters are adaptive, the convergence of the training process slows down due to the large search space. The hybrid

algorithm converges faster by decreasing the search space dimension of the BP algorithm. The hybrid algorithm operates in two stages, forward pass and backward pass. In the forward pass, the premise parameters are set fixed while inputs are injected into the process and the consequent parameters are defined by LSM in the fourth layer. In the backward pass stage, the output error is propagated backward from the output towards input layer. In this stage, while the consequent parameters are set fixed, the premise parameters are updated using BP.

7.3.2.2 Radial Basis Function Network

Radial Basis Function (RBF) network is a type of artificial neural network which uses a radial basis activation function [11]. The RBF network is a feed-forward network model with good performance, global approximation, and is free from local minima problems. An RBF network is a multi-input, single-output system that typically consists of an input layer, a hidden layer with a non-linear RBF activation function, and a linear output layer. The simple architecture of the RBF network is shown in Figure 7.3.

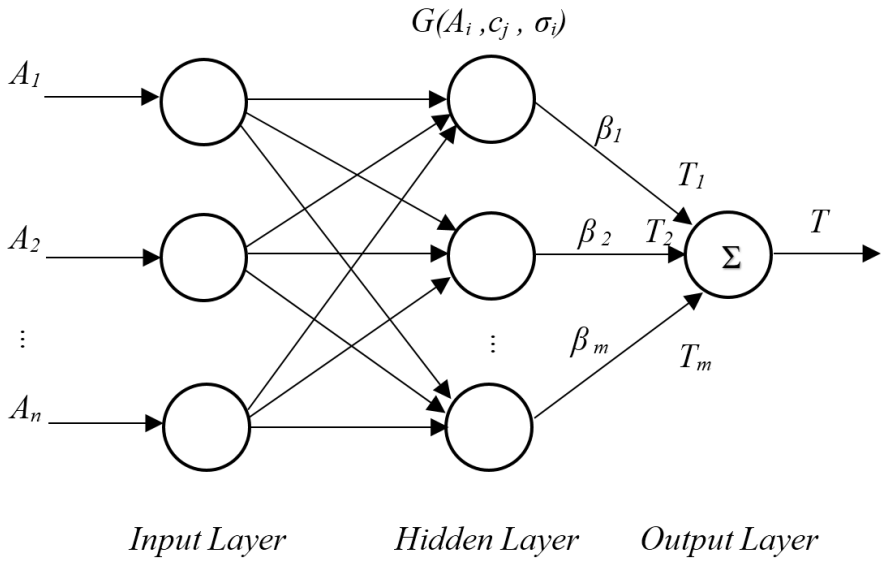


Figure 7.3: RBF network architecture.

A Gaussian function is typically considered as the RBF activation function. The primary problem in a RBF network is to approximate (tune) the centre and width of the Gaussian activation function. The output of each hidden node (Gaussian activation function) is defined as:

$$G(A) = exp\left(-\frac{(A-c_i)^2}{2\sigma_i^2}\right), \qquad i = 1,\ldots,m, \tag{7.12}$$

where in Equation 7.12, A is an input feature vector with n dimension, c_i is the centre, and σ_i is the width of the i_{th} Gaussian function. The output node performs linear summation and is defined as:

$$T(A) = \sum_{i=1}^{m} \beta_i G_i(A), \tag{7.13}$$

RBF network training is supervised and it is performed through providing the input vector of samples and the respected desired outputs for each sample. At each iteration of the RBF network training process, the values of the Gaussian function parameters, number of neurons, and the weights are updated to guarantee that each of the nodes in the hidden layer is properly parametrized and the model error is obtained.

7.4 Experimental Setup

This section describes the experimental setup used in this research to estimate and correct the SVO drift. The first part describes the experimental material setup for this research and the second part presents the structure and setup of the machine learning system as the core methodology to improve the SVO drift in this research.

7.4.1 Experimental setup and materials

The experiments presented in this chapter have been performed in two stages, the first part has been conducted using the data from the KITTI Vision Benchmark Suite [44], while the second part is performed using the collected data by MARIO. The KITTI odometry dataset includes the raw left and right image sequences, ground truth trajectories, and the camera calibration information. As mentioned previously, MARIO is equipped with the ZED stereo camera, 9 DOFs Razor IMU, and a Swiftnav Piksi RTK GPS unit as well. The experiments were also conducted in the open field (Ilam Field, Christchurch, New Zealand). Navigating the robot in this open field provides good satellite coverage for the GPS unit to measure pose information as the ground truth.

7.4.2 Machine learning structure and setup

The experiments were conducted using the KITTI benchmark database and the MARIO database for both training, validation, and testing the machine learning system. For this aim, one set of data was used for each of the stages to train the machine learning system. For each set, the motion was estimated as the output of the SVO system while machine learning parameters were also extracted at each time k. The total number of samples used for training of the first set (KITTI data) was 371, and 1636 samples from a MARIO dataset. Each sample includes the inlier percentage and reprojection error as the input data, and the SVO drift ratio as the output for time k. Figure 7.4 shows a summary of the two datasets used in the experiments at training, validation, and testing stages, including the number of samples (image frames).

Figure 7.5 presents the SVO estimation providing the data from the training datasets in comparison to the ground truth (GPS) data. It is seen that the drift in Z axis is larger than the drift in $X-Y$ axes. The drift ratio quantification provided in Equation 7.6 does not represent

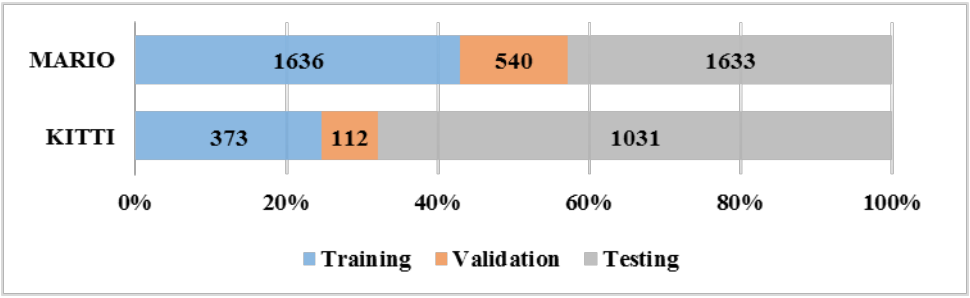


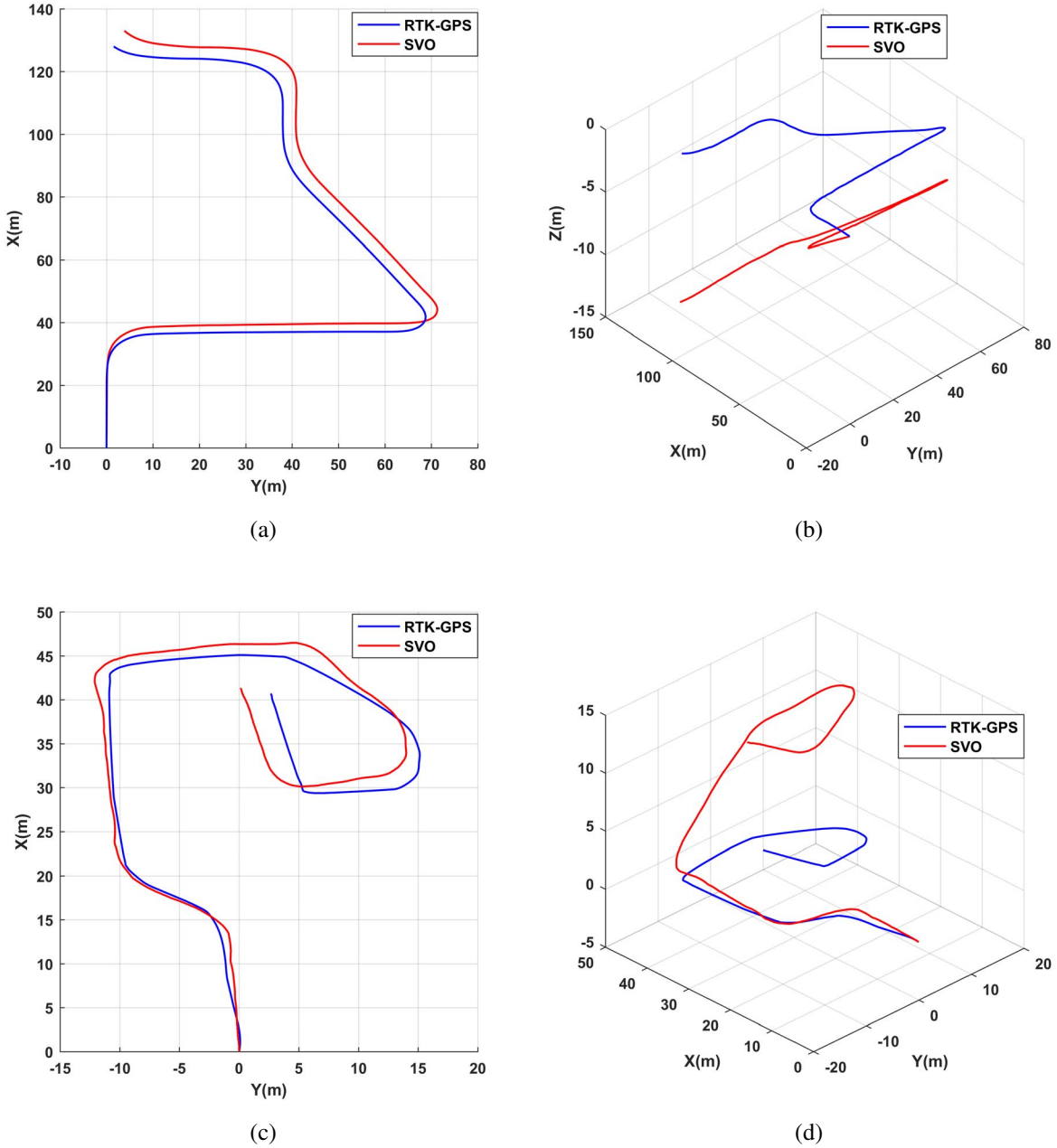
Figure 7.4: Number of samples used for training, validation, and testing.

the right drift ratio to quantify the Z axis drift. For this reason, the drift ratio quantification is formulated independently for Z axis. Thus, one system is trained for $X-Y$ axes drift and one other is trained for the Z axis drift. Both systems are trained with the same training data.

Each of the training datasets are used to train both the ANFIS and RBF network. Figure 7.6 shows the ANFIS structure used in this research. The ANFIS structure for two inputs and one output includes 5 Gaussian membership function for each input, 25 fuzzy rules. All the values are normalized to be in the range of [0 1]. ANFIS and RBF modelling is achieved using MATLAB Neuro-Fuzzy Designer toolbox and NNTOOL [R2016a, Mathworks].

The training process for the first dataset to quantify the $X-Y$ drift is shown in Figure 7.7. As mentioned earlier, the Gaussian membership function parameters including the centre and width are adjusted during ANFIS training process. Figures 7.7c and 7.7d show the adjusted member functions after the training process for input1 and input2 correspondingly. Figure 7.7e shows the 3D surface between the inputs and output samples of the training data.

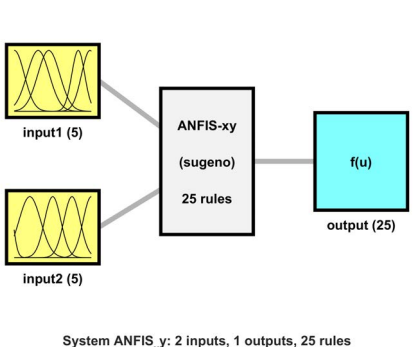
The structure of RBF network is also shown in Figure 7.8 Similarly to ANFIS, the same input and output samples are used to train the RBF network. 373 Gaussian nodes are in the



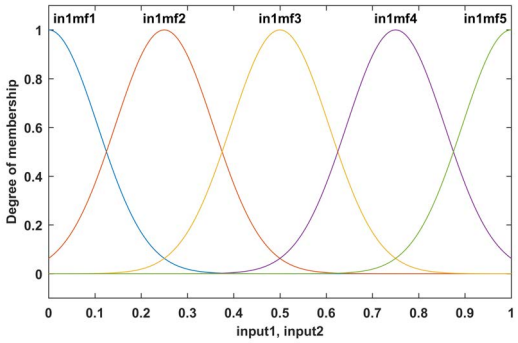
(a) (b)

(c) (d)

Figure 7.5: 2D and 3D plots of the training datasets; (a) and (b): KITTI dataset; (c) and (d): MARIO dataset.



(a)



(b)

Figure 7.6: ANFIS structure: (a) MATLAB modelled ANFIS structure, (b) first and second input initial membership functions.

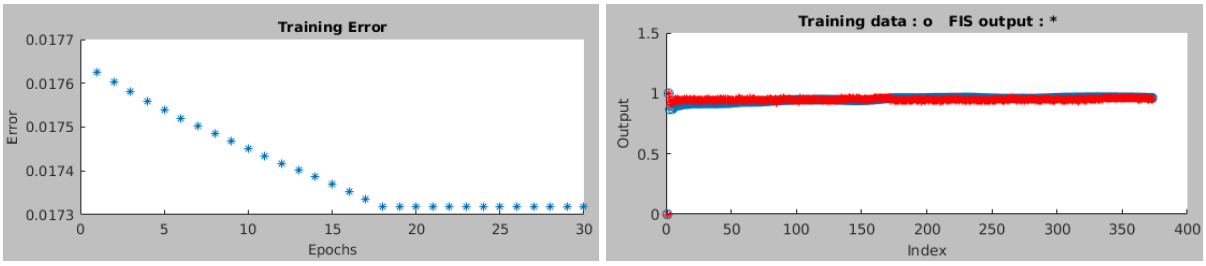
Table 7.2: Estimation error of the trained systems using the training input samples to estimate the SVO drift ratio, ε (m/m).

	ANFIS_XY	ANFIS_Z	RBF_XY	RBF_Z
KITTI	3.0462E-04	4.7E-04	2.9890E-04	4.5E-04
MARIO	1.7351E-04	5.4E-04	1.7352E-04	5.4E-04

RBF layer for the total 373 samples, while the linear output layer includes one node which sums the output of the RBF nodes and results the final output.

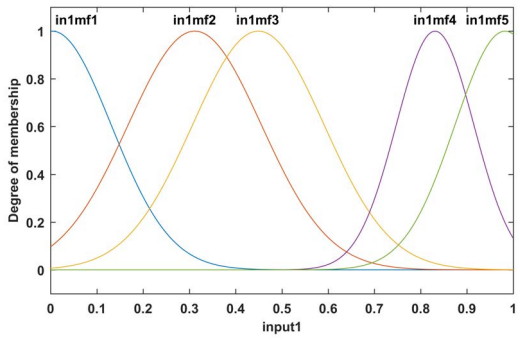
Table 7.2 includes the quantified RMSE of the trained ANFIS and RBF network by simulating them using the training data. The represented values are the error between the estimated and actual SVO drift ratio.

To validate the trained systems, both systems are tested using the samples from a different dataset where the same setup has been used to collect data. The validation process includes one dataset from each KITTI and MARIO databases which the number of samples are about the 30% of the total samples used in the training process. The validation is performed by

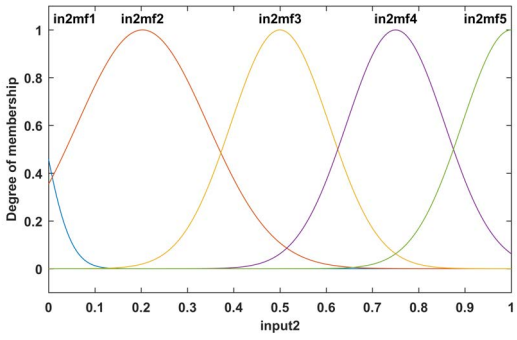


(a)

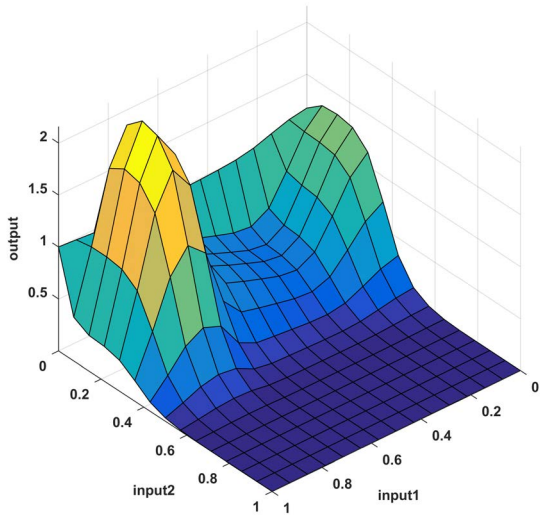
(b)



(c)

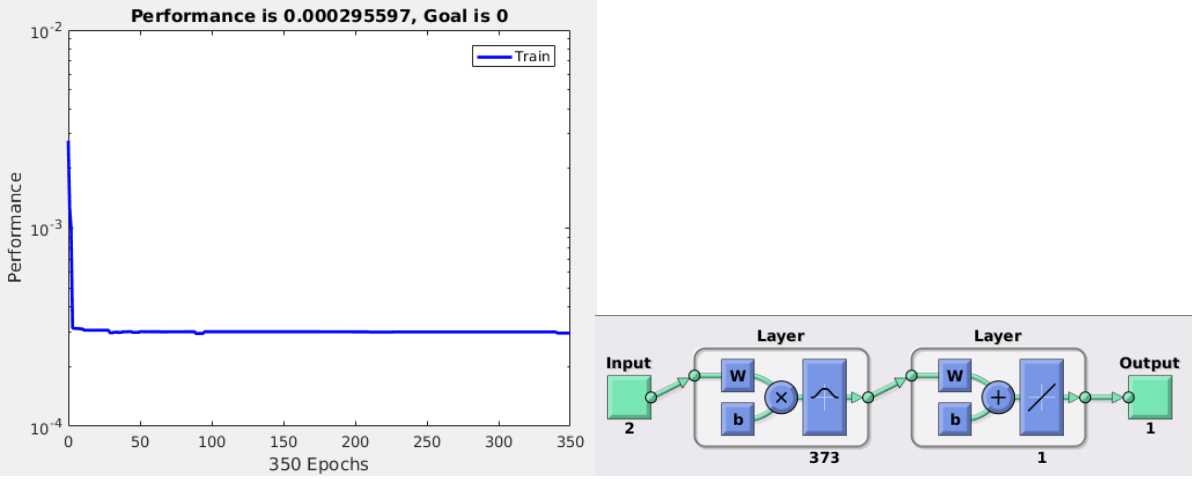


(d)



(e)

Figure 7.7: ANFIS X-Y drift training process on KITTI data: (a) training average error, (b) simulated training data with the trained model, (c) and (d): the tuned first and second input after training, (e) inputs and output surface plot after training.



(a) (b)

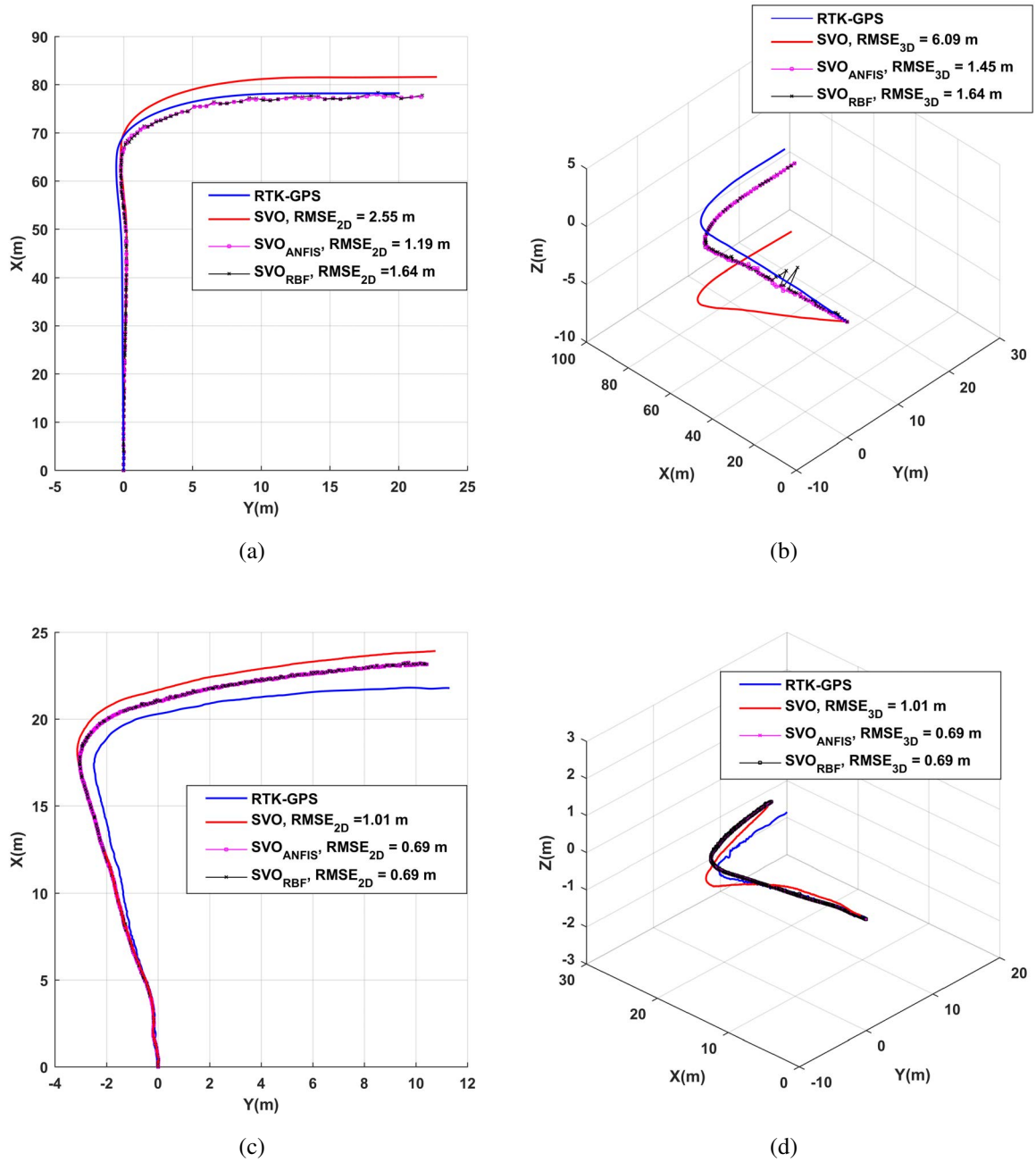
Figure 7.8: RBF network: (a) performance error of X-Y drift training process on KITTI data, (b) RBF network structure.

simulating the $X-Y$ and Z trained systems using the validation data. Both 2D and 3D results for each scenario are presented in Figure 7.9. The validation results for both 2D and 3D scenarios are showing extensive improvements of the SVO drift, while ANFIS has shown better results (lower RMSE) in comparison to the RBF network results on the KITTI validation data while both present the same performance on the MARIO validation data.

7.5 Results and Discussion

The testing results are obtained from the both KITTI and MARIO datasets using the trained ANFIS and RBF systems and have been performed the same way as the validation results were obtained. These results are demonstrated in Figure 7.10, where it presents the 3D results and the associated quantified RMSE error value for each of the tests. From the results it is seen that in all cases, the SVO results of both ANFIS and RBF trained systems have been improved with the average improvement of 50 % in comparison to the original SVO results.

The results presented in Figures 7.10a and 7.10c show the very similar performance of



(a) (b)

(c) (d)

Figure 7.9: Validation 2D and 3D results: (a) and (b) KITTI dataset, (c) and (d) MARIO dataset.

both ANFIS and RBF systems. The two cases presented in Figures 7.10b and 7.10d, show the better performance of ANFIS over the RBF network in form of smaller RMSE, resulted from the bad estimation of RBF in the corners over turning curves. This is due to existence of input parameters out of the normalized range of input data used for the training of ANFIS and RBF systems. In this case, ANFIS shows better performance of being stable to such a noisy data in comparison to the RBF network.

Figure 7.11shows the plotted original and corrected drifts over time from one of the test data (Figure 7.10a). As it can be seen in Figure 7.11, the original drift as the output of the libviso2, increases over time and this increase results a bigger drift at the end of the travelled path. As it has been discussed earlier, the error from the computed transformation matrix at each frame accumulates over time by the matrix concatenation and results a much bigger drift over time. By limiting the error at each frame, the overall drift is corrected and improved. The SVO drift corrections by both ANFIS and RBF systems for MARIO1 test data are presented in Figure 7.11, while the average translational drift in form of RMSE error from *libviso2* is $1.31\,m$, the average translational drift by ANFIS and RBF systems are $0.21\,m$ which is 66% SVO drift improvement. A summary of the results including the average drift and the error is presented in Table 7.3.

Based on the presented summary of the testing results in Table 7.3, Both ANFIS and RBF network were able to significantly improve and reduce the translational drift from the original one from 46% to 70%. Despite the fact that both ANFIS and RBF achieve the same results, ANFIS is considered as the core SVO enhancement method due to the ease of software implementation on MARIO for real-time operation.

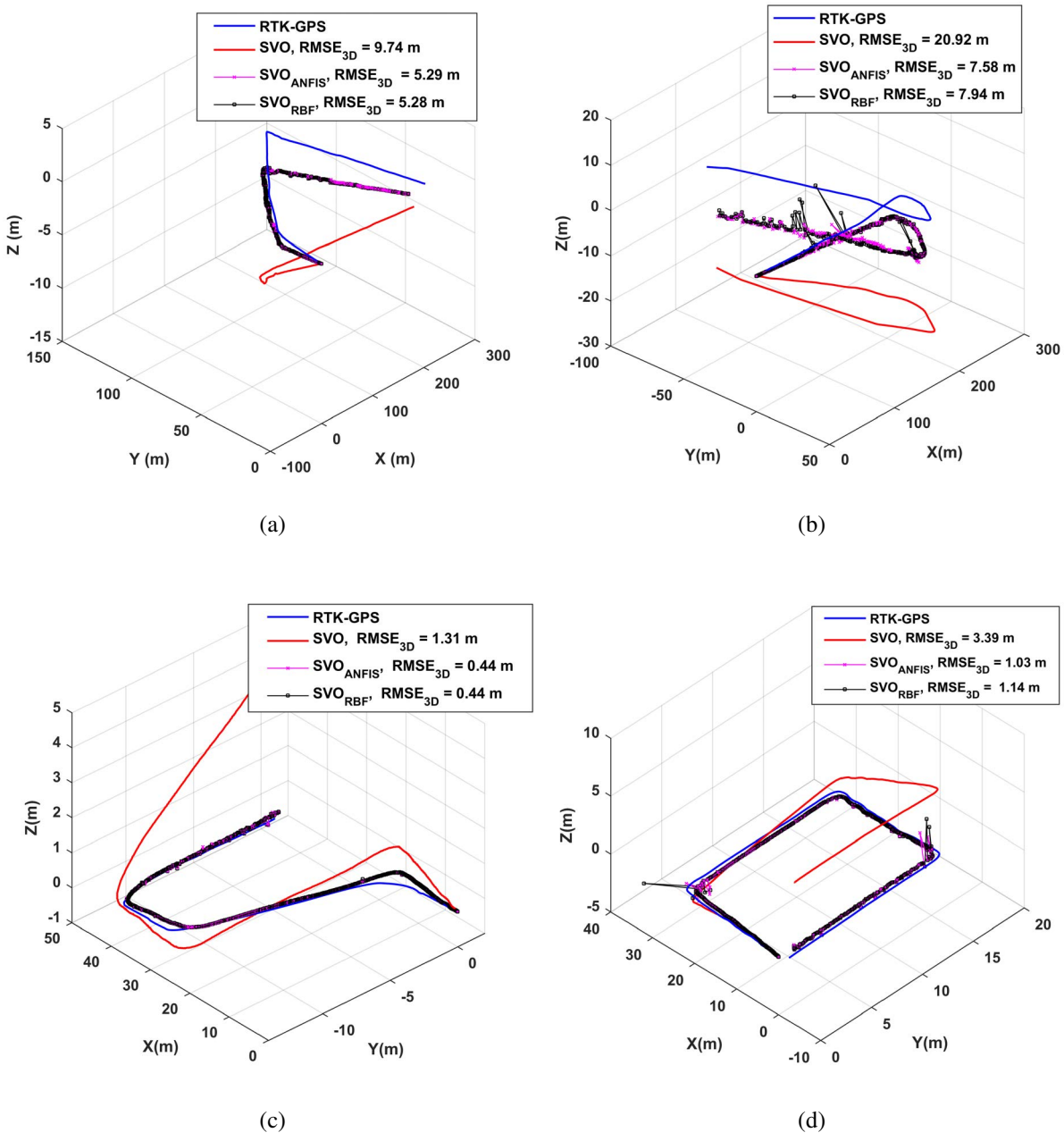


Figure 7.10: Test results: (a) KITTI1 and (b) KITTI2 , (c) MARIO1 and (d) MARIO2.

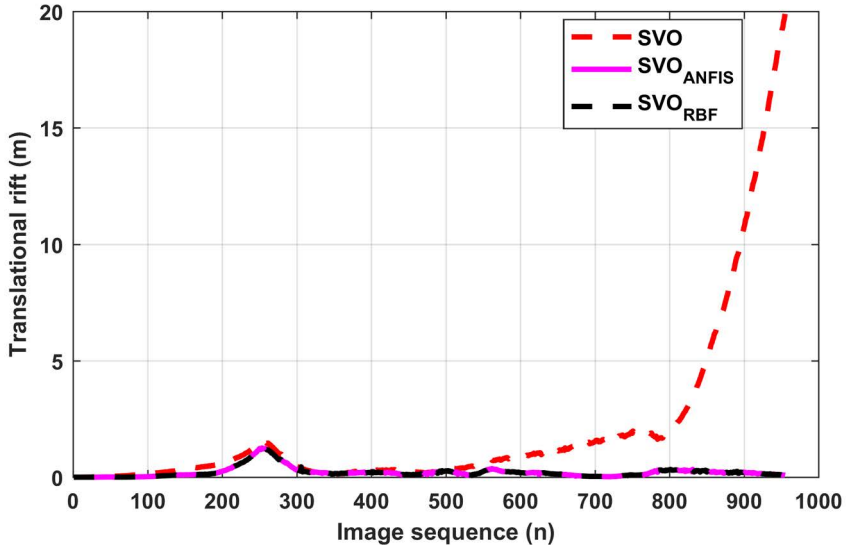


Figure 7.11: The SVO translational Drift over time (MARIO1).

Table 7.3: Summary of the test results presenting the average drift and drift reduction percentage.

Test data	SVO average drift (m)	SVO_{ANFIS} average drift (m)	SVO_{ANFIS} drift reduction %	SVO_{RBF} average drift (m)	SVO_{RBF} drift reduction %
KITTI1	9.74	5.29	45.6	5.28	45.7
KITTI2	20.92	7.58	63.74	7.94	62.05
MARIO1	1.31	0.44	66.48	0.44	66.48
MARIO2	3.39	1.03	69.53	1.14	66.16

7.6 Summary

The work presented in this chapter introduced a machine learning approach to improve the results of SVO by reducing the SVO translational drift, aiming for better localization in GPS-denied environments. The SVO translational drift estimation was defined as a function approximation problem using ANFIS and RBF network. Both ANFIS and RBF network were trained using the samples of the formulated input and output parameters achieved in the SVO process. Two input parameters were 1) inlier percentage, and 2) reprojection error, and the output parameter was the SVO drift ratio. Each system was trained, validated, and tested successfully using the data from the two different datasets including the online KITTI benchmark dataset, and MARIO, the dataset from the experimental equipment used in this research. The results from both datasets presented the improved SVO by reducing the translational drift at each frame and minimizing the overall SVO drift from 46 % to 70 %. The next chapter focuses on utilizing and combining the both developed sensor fusion and machine learning methods in improving the MARIO localization for navigation in GPS-denied environments through the final experimental study and analysis.

Chapter 8

Enhanced Localization for Navigation in GPS-Denied Environments

8.1 Introduction

This chapter presents the approach for enhanced localization in GPS-denied environments. The first novel approach is to evaluate the combination of the two methods previously developed in Chapters 6 and 7 for enhancing the SVO drift, i.e. EKF sensor fusion and ANFIS based drift correction system. For this aim, the enhanced SVO introduced in Chapter 7, is fused with the IMU rotational data using EKF. Therefore with this approach, the SVO translational and rotational drift and in overall, the SVO drift is expected to decrease. The second aim is to investigate the sensor fusion of wheel odometry (WO) as another available source of localization and its influence on the overall results to achieve a reliable localization alternative for navigation of MARIO in GPS-denied environments.

8.2 Methodology

The details of sensor fusion using EKF and ANFIS drift correction system to improve the SVO rotational and translational drift were both described previously in Sections 6.2.2 and 7.3. This chapter investigates combination of the two mentioned methods to achieve the enhanced SVO system, i.e. ANFIS corrected SVO fusion with the IMU data using EKF. Furthermore, sensor fusion of WO with the IMU and ANFIS corrected SVO to achieve a reliable non-GPS localization system is also investigated.

The experiments and analyses have been performed on the recorded experimental data from MARIO in one run. The field experiment for this section was carried out by driving MARIO over a 165m closed loop trajectory on Ilam field while recording all the data using *rosbag* for post processing and analysis. The recorded data includes the stereo image frames, IMU data, WO, RTK-GPS data, and other ROS related data. For each part of the experiment and analysis, the recorded data is played back. The total number of frames acquired by the stereo camera was 1414 frames, so that the data from the other sensors or EKF output with faster rate were resampled to 1414 samples. The ground truth data recorded from the on-board RTK-GPS unit is used for comparisons.

Different following scenarios were tested:

Case (A) SVO vs. SVO_{ANFIS} vs. $SVO_{ANFIS-Filtered}$,

Case (B) SVO fused with IMU vs. $SVO_{ANFIS-Filtered}$ fused with IMU,

Case (C) WO vs. WO fused with IMU;

Case (D) SVO fused with WO and IMU vs. $SVO_{ANFIS-Filtered}$ fused with WO and IMU,

where Case (A) evaluates the performance of the ANFIS corrected SVO in comparison to the original SVO by reiterating the method and analysis performed in Chapter 7, Case (B) evaluates sensor fusion of the both original and corrected SVO estimates with IMU by reiterating the method and analysis performed in Chapter6. Case (B) combines the two methods and aims to evaluate the performance of the final enhanced SVO. Case (C) presents the WO

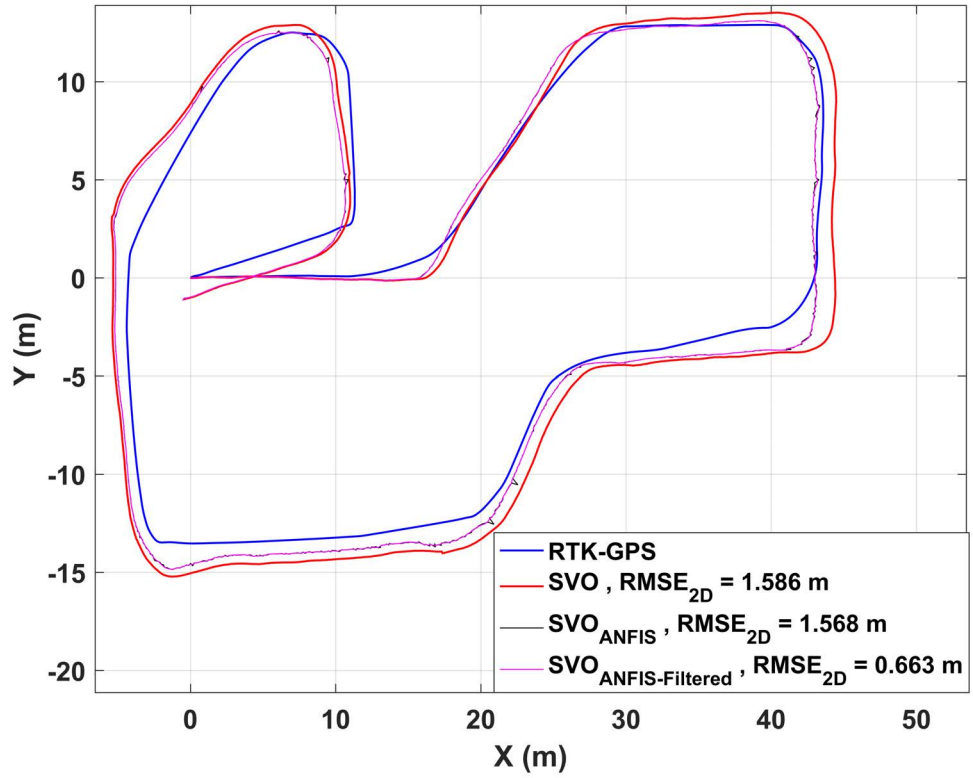
results and its performance. Case (D) aims to evaluate the performance of the sensor fusion of all of the available sources including ANFIS corrected SVO, WO, and IMU, aiming a reliable and lower error non-GPS localization system.

8.3 Experimental Results

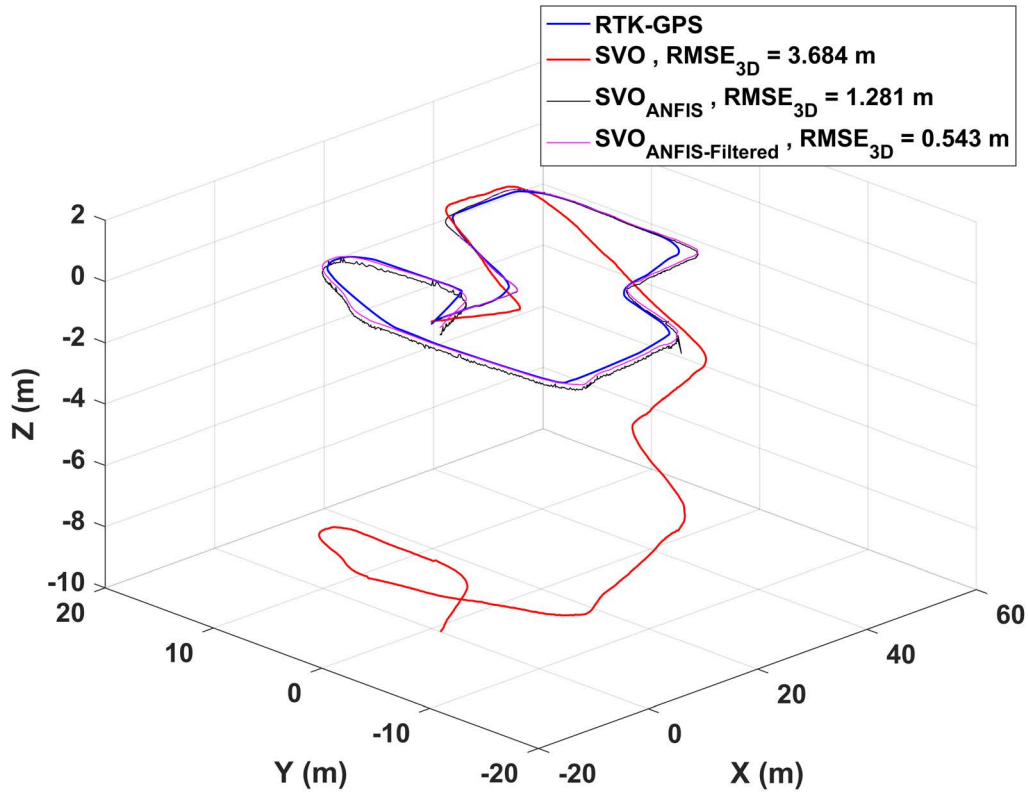
Figure 8.1 shows the results achieved for Case (A) by playing back the stereo image frames for SVO ROS node to get the original SVO and the corrected SVO (SVO_{ANFIS}) in 2D (Figure 8.1a) and 3D (Figure 8.1b) representation. SVO_{ANFIS} estimation is noisy and not as smooth as the original SVO. This causes EKF dysfunction for fusion of the noisy data with another sensor such as IMU. SVO_{ANFIS} is smoothed by one stage of EKF filtering through feeding ($X_k^{SVO_{ANFIS}}$, $Y_k^{SVO_{ANFIS}}$, $Z_k^{SVO_{ANFIS}}$) and generating the filtered output of ($X_k^{SVO_{ANFIS-Filtered}}$, $Y_k^{SVO_{ANFIS-Filtered}}$, $Z_k^{SVO_{ANFIS-Filtered}}$).

In Case (B), both original SVO and $SVO_{ANFIS-Filtered}$ are fused with the IMU data in two different runs. In this regard, EKF integrates the 3D position estimate (X_k^{svo} , Y_k^{svo} , Z_k^{svo}) at each frame from SVO and the 3D orientation ($roll_k^{imu}$, $pitch_k^{imu}$, yaw_k^{imu}) from IMU at time k. EKF handles the synchronisation of data with the different sampling rates. The filtered 6D position estimation data (X_k^{ekf} , Y_k^{ekf} , Z_k^{ekf} ,$roll_k^{ekf}$, $pitch_k^{ekf}$, yaw_k^{ekf}) is reported as the output of EKF. Figure 8.2 shows results of the data fusion for this scenario in 2D (Figure 8.2a) and 3D representation (Figure 8.2b).

In Case (C), WO generated by the MBC forward kinematics from the wheel encoders is also fused with the IMU rotational data. WO 2D pose estimate (X_k^{wo} , Y_k^{wo}) is fused with only (yaw_k^{imu}) of IMU. The results of the original WO estimation and fused IMU and WO

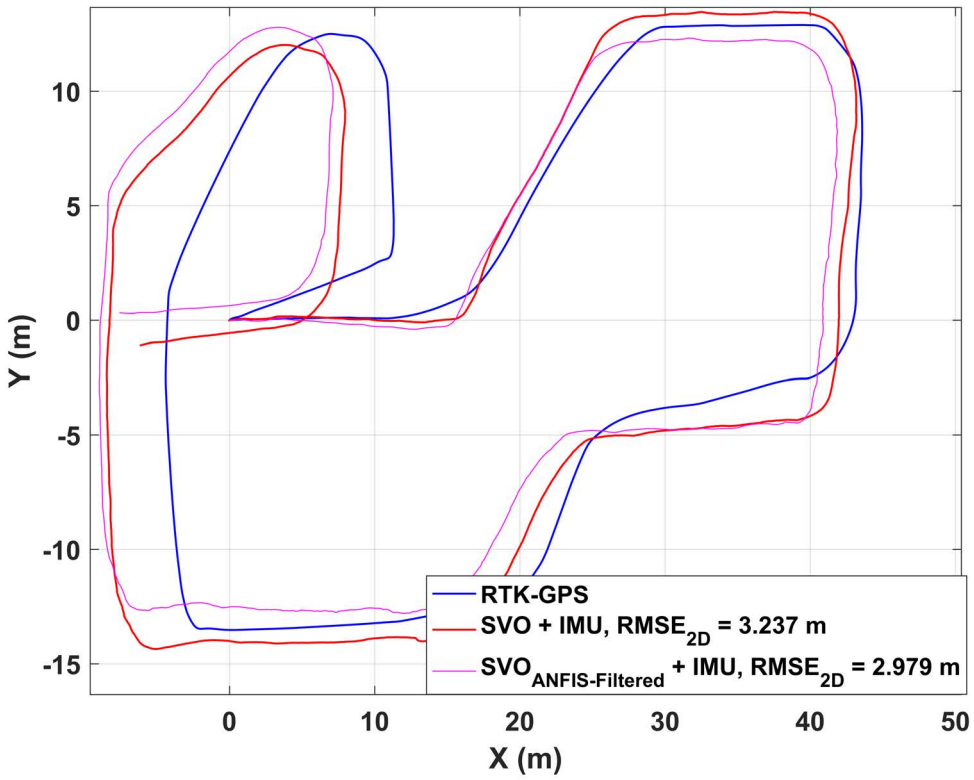


(a)

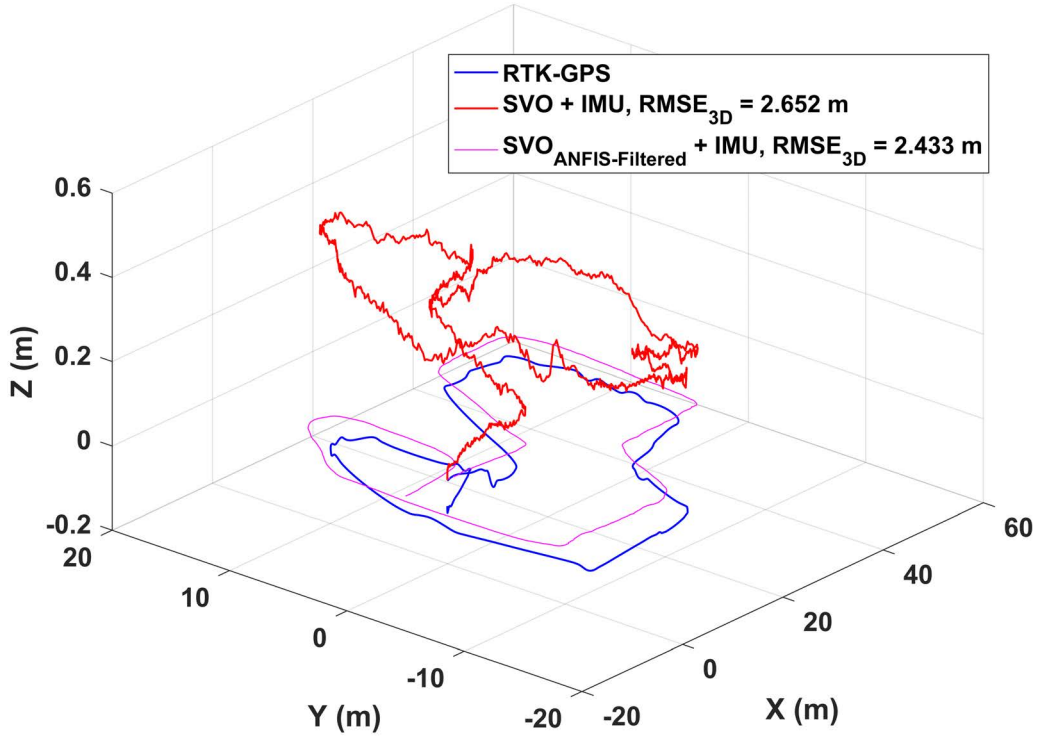


(b)

Figure 8.1: SVO, SVO_{ANFIS} , and $SVO_{ANFIS-Filtered}$ results in comparison to the ground truth from RTK-GPS: (a) X-Y 2D presentation; (b) X-Y-Z 3D representation.



(a)



(b)

Figure 8.2: Fusion of IMU with the original SVO and $SVO_{ANFIS-Filtered}$ in comparison to the ground truth from RTK-GPS: (a) X-Y 2D presentation; (b) X-Y-Z 3D representation.

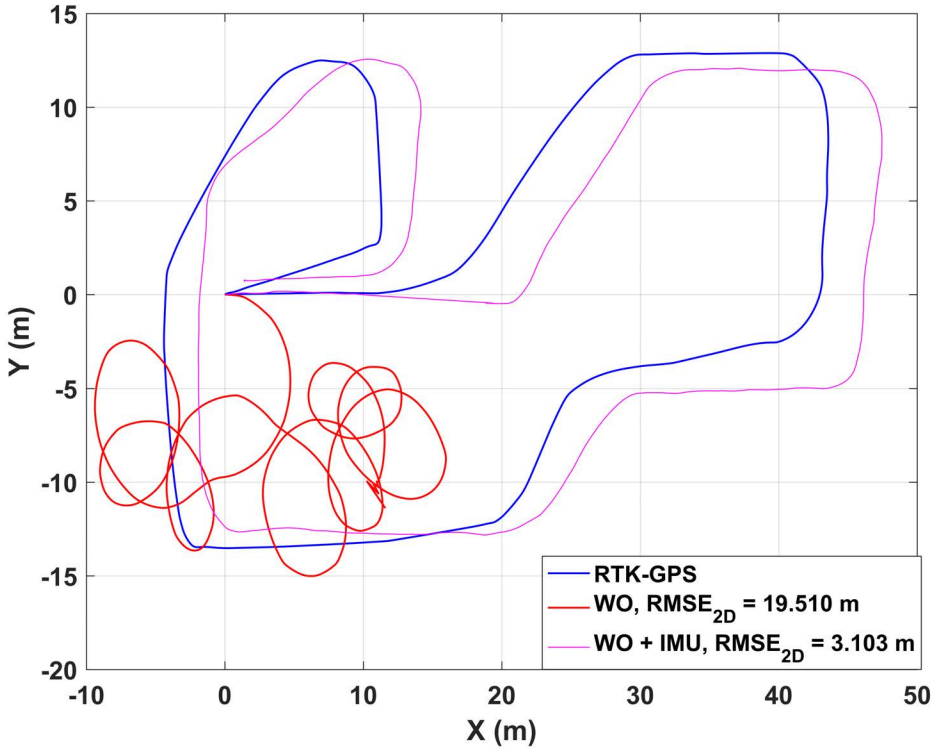
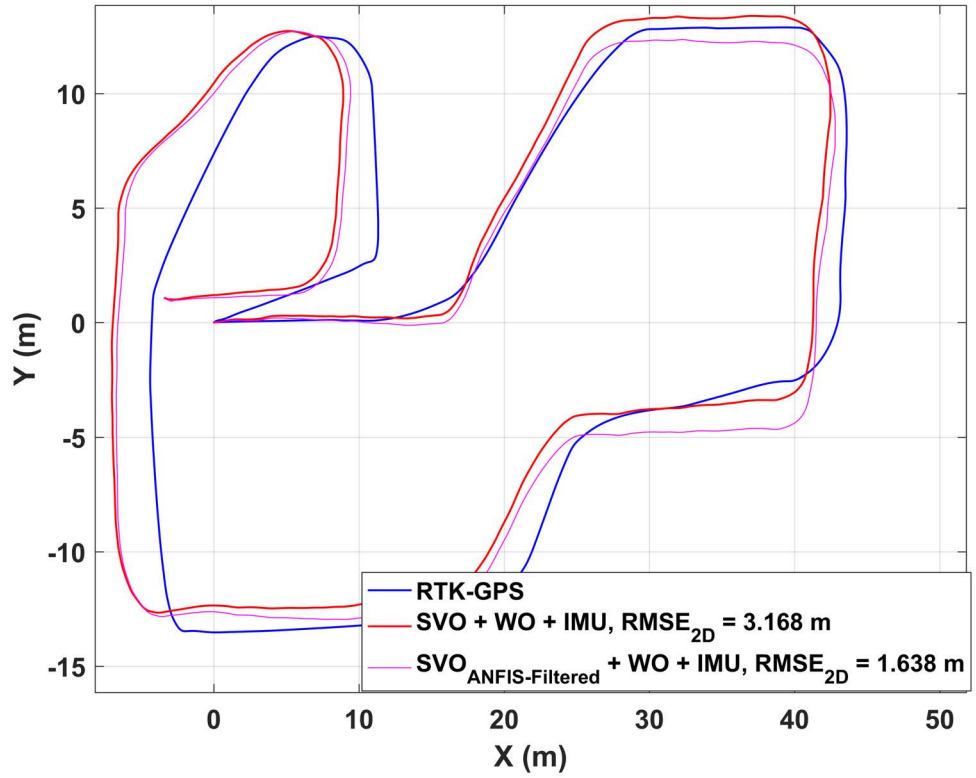


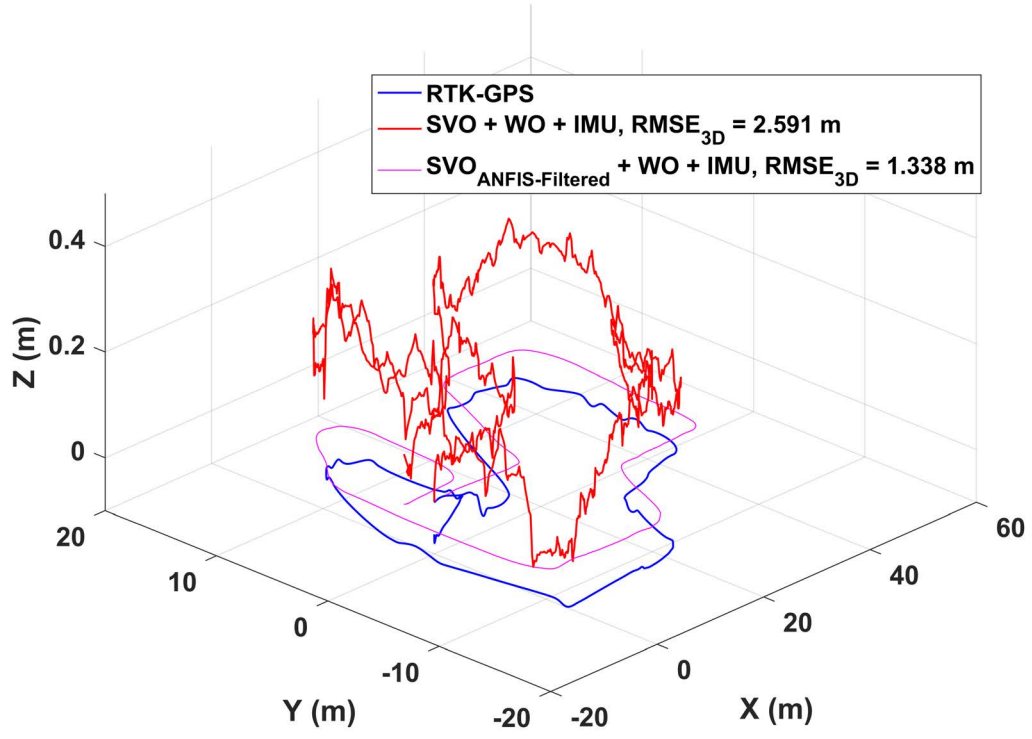
Figure 8.3: WO and the fusion of IMU with WO in comparison to the ground truth from RTK-GPS in X-Y 2D presentation.

are presented in Figure 8.3. Unfortunately the rotational estimation of WO has failed for this run of field experiment but the fused (WO + IMU) data just requires the 2D translational information and relies on the IMU (yaw_k^{imu}).

Case (D) aims to evaluate the performance of the case involving integration of the data from all the available sources for a reliable and accurate localization in GPS-denied environment. To achieve this, SVO 3D position estimate (X_k^{svo} , Y_k^{svo} , Z_k^{svo}), WO 2D position estimate (X_k^{wo} , Y_k^{wo}) , and IMU 3D orientation ($roll_k^{imu}$, $pitch_k^{imu}$, yaw_k^{imu}) at time k are integrated and the EKF 6D position and orientation estimate (X_k^{ekf} , Y_k^{ekf} , Z_k^{ekf} ,$roll_k^{ekf}$, $pitch_k^{ekf}$, yaw_k^{ekf}) is obtained as the integrated output. Figure 8.4 shows the results of integration for both SVO + WO + IMU and $SVO_{ANFIS-Filtered}$ + WO + IMU cases.



(a)



(b)

Figure 8.4: Fusion of IMU and WO with the original SVO and $SVO_{ANFIS-Filtered}$ in comparison to the ground truth from RTK-GPS: (a) X-Y 2D presentation; (b) X-Y-Z 3D representation.

Table 8.1: Error quantification of the results from the different parts of the experiment.

Experiment parts	***2D drift*** *(m)*	***2D error*** *(%)*	***3D drift*** *(m)*	***3D error*** *(%)*	***Max drift*** *(m)*	***Max error*** *(%)*
SVO	1.58	0.95	3.68	2.22	9.94	6.02
SVO_{ANFIS}	1.56	0.94	1.28	0.77	4.85	2.93
$SVO_{ANFIS-Filtered}$	0.66	0.4	0.54	0.32	2.35	1.42
WO	19.51	11.8	15.93	9.63	55.55	33.66
SVO + IMU	3.23	1.95	2.65	1.6	8.2	4.96
$SVO_{ANFIS-Filtered}$ ***+ IMU***	2.97	1.8	2.43	1.47	7.58	4.59
WO + IMU	3.10	1.87	2.53	1.53	8.4	5.09
SVO + WO + IMU	3.16	1.91	2.59	1.56	7.83	4.74
$SVO_{ANFIS-Filtered}$ ***+ WO + IMU***	1.63	0.99	1.33	0.8	3.77	2.28

Table 8.1 provides the overall error quantification results for each part of the experiment in form of 2D and 3D average drift and average error percentage in comparison to the ground truth positioning data from the RTK-GPS. The average drift is formulated in form of RMSE between the test results and the ground truth data. It also presents the error in terms of maximum drift and maximum error for each case. The average 2D and 3D error is also shown as a bar chart in Figure 8.5.

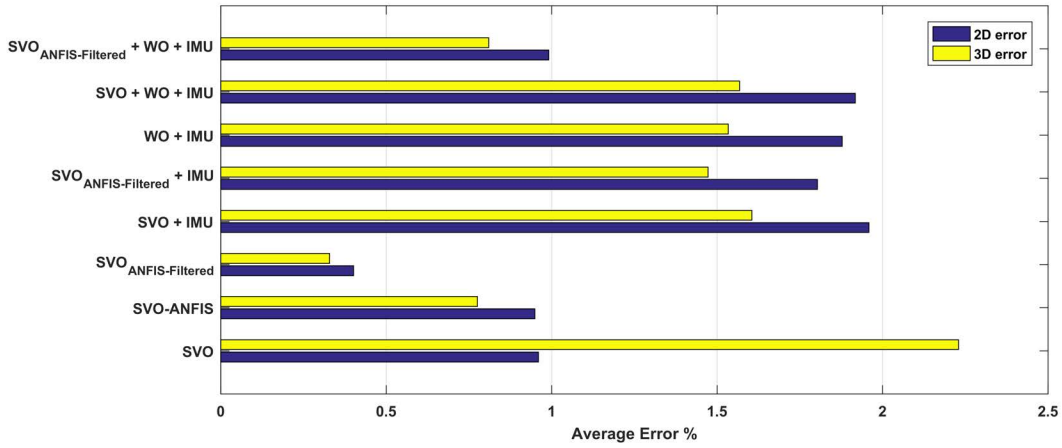


Figure 8.5: Bar chart representation of the results errors from the different parts of the experiment.

8.4 Discussion and Analysis

In Case (A), the original SVO and SVO_{ANFIS} results depicted in Figure 8.1 present better performance of SVO_{ANFIS} as expected, specifically in 3D form by reducing the elevation drift and as a results, the overall drift to 0.77% in comparison to the original SVO error 2.22%. As mentioned earlier, SVO_{ANFIS} was filtered due to effect of noisy estimate on the EKF performance. As a result, $SVO_{ANFIS-Filtered}$ shows better performance by the average error of 0.32%. The results for this part revalidates the effectiveness of ANFIS drift correction system developed in Chapter 7 to obtain an enhanced SVO estimation.

Case (B) investigates benefits of the sensor fusion to integrate $SVO_{ANFIS-Filtered}$ and IMU data. With this experiment, it is desired to show if and how the fusion of $SVO_{ANFIS-Filtered}$ translational data and IMU rotational data improves the overall drift by both improving the translational and rotational drifts. The results were presented in Figure 8.2. The fusion of $SVO_{ANFIS-Filtered}$ and IMU, results average error of 1.8% in 2D and 1.47% in 3D while the fusion of original SVO and IMU, results 1.95% and 1.06% correspondingly. Fusion of

$SVO_{ANFIS-Filtered}$ and IMU shows a better performance in error reduction in comparison to the fusion of SVO and IMU as expected, but bigger error in comparison to the $SVO_{ANFIS-Filtered}$ results which is not expected. On other hand, the 2D error of fused SVO and IMU is also bigger than the original error, while the 3D error shows better performance due to substantial reduction of the SVO elevation. One of the main reasons to describe this behaviour might be the EKF parameters adjustment. EKF parameters such as process noise covariance matrix is set to the default values, while it needs a proper dynamic tuning to achieve the best performance out of EKF. However, EKF performance enhancement is not the focus of this work and can be considered as the future work of this thesis.

In a reliable non-GPS localization approach, relying only on one source of positioning data is risky. The reliability term matters when one of the data sources is unavailable, so that the robot pose can be estimated through the other input sources. Thus, the integration of all of the available positioning and inertial data is advisable. As the next part of the experiment, fusion of WO as another available source of relative positioning data with IMU and SVO was investigated. For Case (C), the original WO estimation shown in Figure 8.3 does not show a good performance of pose estimation as expected, but the WO and IMU fusion improves the results to a good accuracy of 1.87%. From the results, the original WO orientation estimation has failed due to the existence of a large bias in the WO orientation estimate, however the fusion of WO translation and IMU orientation achieves low error results in the range of SVO error. WO translational drift is also seen in the results due to wheel slippage which is unavoidable.

The results for the last part of the experiment in Case (D) involving fusion of the data from all the sources, aiming a reliable and accurate localization, were shown in 8.4. The results from this part show improvement of the drift and error in comparison to all of the other parts, where partial sensor fusion was tested, i.e. SVO + IMU, $SVO_{ANFIS-Filtered}$ + IMU, and WO + IMU. The fusion of $SVO_{ANFIS-Filtered}$, WO, and IMU achieves better results by reducing the error in about 50% in comparison to the case where SVO, WO, and IMU are fused. In this

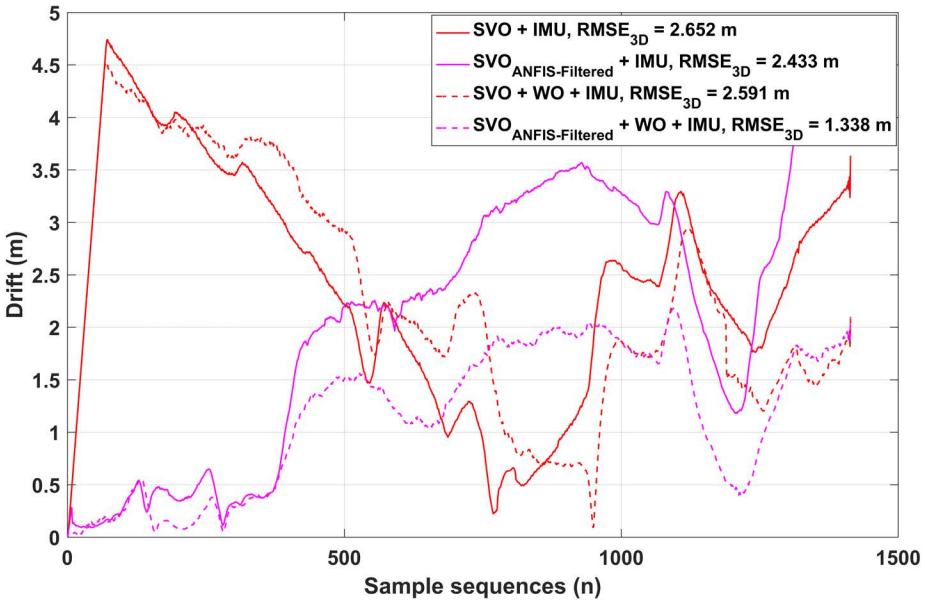


Figure 8.6: Drift trends of the fused results from the different parts of the experiment over the experiment run.

regard, the 2D error is reduced from 1.91% to 0.99%, while the 3D error has a decay from 1.56% to the 0.8%.

Figure 8.6 shows the trend of drift over time for four different parts of the experiment. As shown, fused $SVO_{ANFIS-Filtered}$, WO, and IMU achieves the best results among all the other cases, while fusion of only $SVO_{ANFIS-Filtered}$ and IMU also achieves better results in comparison to the fusion of SVO,WO, and IMU.

A comparison between $SVO_{ANFIS-Filtered}$, fused $SVO_{ANFIS-Filtered}$ and IMU, and fused $SVO_{ANFIS-Filtered}$,IMU, and WO depicted by Figure 8.7 shows better results of $SVO_{ANFIS-Filtered}$ among the three of them. However as discussed, better results do not guarantee a reliable solution specifically for outdoor GPS denied environments. The optimal solution involving the fusion of all data sources achieves the best results so far out of all of the other data fusion configurations. The experimental results achieves the goal of this experiment. The proposed solution can operate as a reliable alternative to the GPS based localization system for a mobile

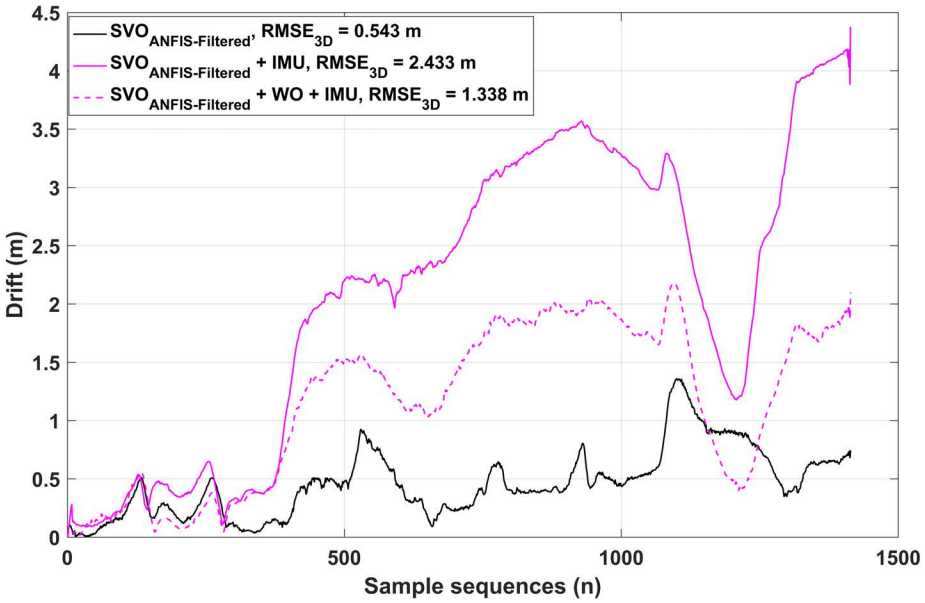


Figure 8.7: Drift trends of the $SVO_{ANFIS-Filtered}$, fused $SVO_{ANFIS-Filtered}$ and IMU, and fused $SVO_{ANFIS-Filtered}$,WO, and IMU estimations over the experiment run.

robot, where the GPS data is lost and the mobile robot needs the state of the system to reliably navigate.

Defining the acceptable error rate for a non-GPS localization system depends on the application. While a general case cannot be considered, we can consider a specific case such as the operation of MARIO in a typical apple orchard. If a $0.5\,m$ wide robot navigates within a standard apple orchard with 100m long rows spaced $3\,m$ apart, a drift of$1\,m$ over $100\,m$ (1% error) at the end of the row can be considered as the acceptable error to avoid collision with the trees. It is assumed that the GPS is not available within the $100\,m$ long row, but will be available during the turns.

8.5 Summary

The experimental work and analysis presented in this chapter, investigates:

1. The effect of the combination of the presented approaches in Chapter 6 and 7, in order

to achieve a more accurate SVO,

2. The effect of sensor data fusion to achieve a reliable and accurate pose estimate as an alternative to the GPS for localization in outdoor GPS denied environments.

The fusion of ANFIS based drift correction and IMU data does not achieve a more accurate result compared to the ANFIS corrected SVO, but it certainly performs better in comparison to the case where the original SVO is fused with IMU. Data fusion of the pose estimates from different sources not only provides a more reliable estimation of the robot's pose for GPS-Denied environments, but through the experimental work presented in this chapter, achieved a more accurate results in comparison to the case where only one or two sources of data was used. Furthermore, the final case including the $SVO_{ANFIS-Filtered}$ **+ *WO* + *IMU*** also satisfied the hypothetical acceptable error rate of 1% in both 2D and 3D case with 0.99% and 0.8% error.

Chapter 9

Conclusion and Future Works

This Chapter summarizes the contributions of this thesis and presents the conclusions of this research. Recommendations for the future works are also provided.

9.1 Conclusion

The work presented in this thesis contributes to the mechatronic design, development, and control of a non-holonomic omnidirectional mobile robotic platform, MARIO, for potential applications in agriculture, alongside the development of an enhanced vision based localization system, enabling navigation in GPS-denied environments such as orchards.

Development of MARIO was the initial part of this thesis that involved mechatronic system design and development of a 4WD4S mobile robot as a non-holonomic omnidirectional platform. The design was processed through an innovative integrated application of CAD/CAM/CAE and RP for rapid development of the robot. 3D CAD design not only allowed fabrication of the platform through CAM and RP, but it also enabled further CAE analysis including structural analysis of the robot parts and motion analysis for torque validation of the active joints. ROS provided the software development environment for interfacing the electronics, sensory system, and the actuation for control and navigation of the robot. With this innovative integ-

rated design approach, successful prototyping of MARIO was achieved with a reasonably low development time and cost.

3D modelling and simulation of MARIO as a non-holonomic omnidirectional mobile robot was essential for off-line programming and validation of the key software components of the platform, specifically the kinematic model-based controller (MBC). MARIO 3D modelling and simulation was achieved using Gazebo simulator. Gazebo simulator also provided simulation of the world environment, sensors, and the joint control interfaces. The MBC formulation including the singularity problem associated with MBC in two forms of mathematical and kinematic singularities was discussed. Mathematical singularities were solved by presenting the idea of switching between the different steering modes. The MBC was tested and validated on both simulated and the physical model. The ROS/Gazebo approach enabled successful development and test of MARIO and different implemented robotic software in a simulation environment before implementation on the real system.

To overcome the kinematic singularity of MBC, A novel fuzzy logic based system was designed and aimed to direct the ICR in a way to avoid a defined singular region around the steering joints axes. The fuzzy control system controlled and directed the ICR on the border of the defined singular region, in case of the placement or pass through the singular region. The simulation and experimental results proved the practicality and usefulness of the proposed method in handling the kinematic singularity on MARIO.

For navigation in agricultural environments such as orchards, a reliable and accurate localization system is essential, specially in the absence of GPS data. Vision based localization for navigation of MARIO in GPS-denied environments was utilised. SVO suffers from the accumulative drift over time in the long range. To improve the SVO performance, drift in form of translational and rotational was identified through the experimental study. To overcome the rotational error, sensor fusion using EKF was employed to integrate IMU rotational data and the SVO translational data. This improved the results as it reduced the rotational drift, con-

sequently minimized the overall drift. However, the SVO translational drift was still evident which could be more improved.

A novel machine learning approach was introduced to improve the SVO accuracy by reducing the SVO translational drift. The SVO translational drift correction system was modelled as a function approximation problem using ANFIS and RBF network. Both ANFIS and RBF network were trained supervised, using the training dataset. Training dataset included samples of the formulated input and output parameters. Two input parameters were 1) inlier percentage, and 2) reprojection error, and the output parameter was the SVO drift ratio. Both trained ANFIS and RBF were tested and validated successfully using the validation and test datasets. Datasets were obtained through recording data with MARIO in different runs on the field. A different group of datasets from the online KITTI benchmark database were also used, where the training, validation, and testing were also successful. The estimated drift ratio based on the two inputs at each frame, could correct and suppress the drift at that frame. Therefore reducing the translational drift at each frame minimized the overall SVO drift from 46 % to 70 %. Both RBF and ANFIS systems presented the same performance but ANFIS was selected as the core drift correction system, due to the ease of implementation for real time processing.

An enhanced vision-based localization system was aimed through utilizing and combining the both sensor fusion and machine learning methods in reducing the SVO drift through the final experimental study and analysis. Sensor fusion of IMU and ANFIS corrected SVO lowered the error in comparison to the fusion of SVO with IMU, but it did not produce better results in comparison to the results from the ANFIS corrected SVO. Furthermore, the integration of WO with the enhanced SVO and IMU was also investigated to achieve a reliable non-GPS localization solution for MARIO provided by the integration of all the available sensory data. Sensor fusion of IMU, ANFIS corrected SVO, and WO not only could provide a more reliable estimation of the MARIO's pose, it also achieved more accurate results in comparison to the cases where only one or two sources of data were used.

9.2 Future Works

The following recommendations are presented for the extension of the current work. They include two major aspects of improvements. The first part is regarding the control system, while the second part is around the challenges for the navigation system.

The current control system (MBC) is based on the kinematic model of the platform. This was due to the fast implementation of the control system to reach to the next stage of the research. For such a mobile robotic system, there are always uncertainties and time-varying parameters which turn the system into a non-linear system. Other external factors such as soil and wind conditions also affect the robot dynamics [78]. These mentioned internal and external factors lead into need for a design of an optimal control system for such a non-linear system. Such a system requires more mathematically complex model and each control function should be modelled explicitly. To provide robust control, different approaches have been proposed and implemented from fuzzy control [58] and adaptive control [29] to sliding mode control (SMC) [34]. Compared to other approaches, SMC has received more attention due to its fast and robust response especially for off-road and unstructured environment. Combination, improvement, and evaluation of these methods can be investigated for MARIO to provide a robust and stable motion control system.

Another potential area of research to improve the current work is the enhancement of the sensor fusion system. For this work, EKF algorithm from ROS robot_localization package provided sensor fusion, however challenges still remain for an optimally tuned EKF system. The experimental results were achieved with the core basic required configuration of the EKF out of many other effective parameters. The current filter does not fuse the linear acceleration as it has not been included in the state estimator and the kinematic model of the filter. This can be formulated and added to improve the EKF accuracy. Furthermore, the filter is set with a default process noise covariance matrix. Finding the optimal process noise covariances is very critical to achieve the optimal performance of the filter. An adaptive online system to define

and tune the EKF process noise would improve the performance the current filter.

The basic navigation of MARIO using ROS navigation stack was possible for semi-autonomous operations, however it was not the focus of this work. The navigation system requires a good estimation of the system state which was mainly focused in this work. Therefore, the challenges are still remained for the future to achieve a reliable navigation. The basic ROS navigation stack provides a 2D semi-autonomous control system for the ground robots in a provided map. SLAM can be employed for map building and localization in the environment using the current stereo camera. Obstacle avoidance system including the associated challenges is another key component of the navigation system which needs to be implemented.

The implementation of 2D or 3D LiDARs can also be investigated in the future for the purpose of pose estimation, SLAM, and obstacle avoidance. The performance of the vision based systems varies based on the light variations in the environment, while LiDARs are not sensitive to the light variations. However, utilizing a LiDAR unit adds technical and research challenges especially for 3D mapping due to the need for a large computational capacity to process the point clouds and the poor performance in unstructured environments.

References

[1] Mohammad OA Aqel, Mohammad H Marhaban, M Iqbal Saripan, and Napsiah Bt Ismail. Adaptive-search template matching technique based on vehicle acceleration for monocular visual odometry system. *IEEJ Transactions on Electrical and Electronic Engineering*, 11(6):739–752, 2016.

[2] Fabio Bellavia, Marco Fanfani, and Carlo Colombo. Selective visual odometry for accurate AUV localization. *Autonomous Robots*, 41(1):133–143, 2017.

[3] Alessandro Benini, Adriano Mancini, and Sauro Longhi. An IMU/UWB/vision-based extended kalman filter for mini-UAV localization in indoor environment using 802.15. 4a wireless sensor network. *Journal of Intelligent & Robotic Systems*, pages 1–16, 2013.

[4] Marcel Bergerman, Silvio M Maeta, Ji Zhang, Gustavo M Freitas, Bradley Hamner, Sanjiv Singh, and George Kantor. Robot farmers: Autonomous orchard vehicles help tree fruit production. *IEEE Robotics & Automation Magazine*, 22(1):54–63, 2015.

[5] Kathryn B Bicknell, Richard J Ball, Ross Cullen, and Hugh R Bigsby. New methodology for the ecological footprint with an application to the New Zealand economy. *Ecological economics*, 27(2):149–160, 1998.

[6] J Blumrich. Omnidirectional wheel, February 5 1974. US Patent 3,789,947.

[7] F Bonaccorso, L Cantelli, D Longo, D Melita, G Muscato, and M Prestifilippo. The U-Go robot, a multifunction rough terrain outdoor tracked vehicle for R&D on autonomous navigation algorithms. In *Proceedings of the 5th IARP Workshop on Robots for Risky Interventions and Environmental Surveillance-Maintenance, Leuven (B)*, pages 20–22, 2011.

[8] Francisco Bonin-Font, Alberto Ortiz, and Gabriel Oliver. Visual navigation for mobile robots: A survey. *Journal of intelligent and robotic systems*, 53(3):263, 2008.

[9] Christian Brenneke, Oliver Wulf, and Bernardo Wagner. Using 3d laser range data for slam in outdoor environments. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, volume 1, pages 188–193. IEEE, 2003.

[10] Claus Brenner. Vehicle localization using landmarks obtained by a lidar mobile mapping system. *Int. Arch. Photogramm. Remote Sens*, 38:139–144, 2010.

[11] David S Broomhead and David Lowe. Radial basis functions, multi-variable functional interpolation and adaptive networks. Technical report, DTIC Document, 1988.

[12] L Bruzzone and G Quaglia. Review article: locomotion systems for ground mobile robots in unstructured environments. *Mechanical Sciences*, 3(2):49–62, 2012.

[13] Fernando Caballero, Luis Merino, Joaquin Ferruz, and Anibal Ollero. Vision-based odometry and SLAM for medium and high altitude flying UAVs. In *Unmanned Aircraft Systems*, pages 137–161. Springer, 2008.

[14] Faruk Caglar, Shashank Shekhar, Aniruddha Gokhale, Satabdi Basu, Tazrian Rafi, John Kinnebrew, and Gautam Biswas. Cloud-hosted simulation-as-a-service for high school stem education. *Simulation Modelling Practice and Theory*, 58:255–273, 2015.

[15] Guy Campion, Georges Bastin, and Brigitte Dandrea-Novel. Structural properties and classification of kinematic and dynamic models of wheeled mobile robots. *IEEE Transactions on robotics and automation*, 12(1):47–62, 1996.

[16] Francesco Capezio, Antonio Sgorbissa, and Renato Zaccaria. Gps-based localization for a surveillance UGV in outdoor areas. In *Proceedings of the Fifth International Workshop on Robot Motion and Control RoMoCo 2005)*, pages 157–162. IEEE, 2005.

[17] Luis Rodolfo García Carrillo, Alejandro Enrique Dzul López, Rogelio Lozano, and Claude Pégard. Combining stereo vision and inertial navigation system for a quad-rotor UAV. *Journal of Intelligent & Robotic Systems*, 65(1-4):373–387, 2012.

[18] Chung L Chang and Jia H Jhu. Zigbee-assisted mobile robot gardener. In *CACS International Automatic Control Conference (CACS)*, pages 41–46. IEEE, 2013.

[19] Yang Cheng, Mark W Maimone, and Larry Matthies. Visual odometry on the mars exploration rovers-a tool to ensure accurate driving and science imaging. *IEEE Robotics & Automation Magazine*, 13(2):54–62, 2006.

[20] Bong-Su Cho, Woosung Moon, Woo-Jin Seo, and Kwang-Ryul Baek. A study on localization of the mobile robot using inertial sensors and wheel revolutions. In *International Conference on Intelligent Robotics and Applications*, pages 575–583. Springer, 2011.

[21] Sunglok Choi, Jaehyun Park, and Wonpil Yu. Simplified epipolar geometry for real-time monocular visual odometry on roads. *International Journal of Control, Automation and Systems*, 13(6):1454–1464, 2015.

[22] Wenyan Ci and Yingping Huang. A robust method for ego-motion estimation in urban environment using stereo camera. *Sensors*, 16(10):1704, 2016.

[23] Christian Connette, Martin Hägele, and Alexander Verl. Singularity-free state-space representation for non-holonomic, omnidirectional undercarriages by means of co-

ordinate switching. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4959–4965. IEEE, 2012.

[24] Christian P Connette, Andreas Pott, Martin Hagele, and Alexander Verl. Control of an pseudo-omnidirectional, non-holonomic, mobile robot based on an ICM representation in spherical coordinates. In *47th IEEE Conference on Decision and Control (CDC 2008)*, pages 4976–4983. IEEE, 2008.

[25] Christian P Connette, Christopher Parlitz, Martin Hagele, and Alexander Verl. Singularity avoidance for over-actuated, pseudo-omnidirectional, wheeled mobile robots. In *IEEE International Conference on Robotics and Automation (ICRA 2009)*, pages 4124–4130. IEEE, 2009.

[26] Peter Corke. Matlab toolboxes: robotics and vision for students and teachers. *IEEE Robotics & Automation Magazine*, 14(4), 2007.

[27] Maria P Diago, Javier Tardaguila, et al. A new robot for vineyard monitoring. *Wine & Viticulture Journal*, 30(3):38, 2015.

[28] Alexander Dietrich, Thomas Wimböck, Alin Albu-Schäffer, and Gerd Hirzinger. Singularity avoidance for nonholonomic, omnidirectional wheeled mobile platforms with variable footprint. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 6136–6142. IEEE, 2011.

[29] Warren E Dixon, Marcio S de Queiroz, Darren M Dawson, and Terrance J Flynn. Adaptive tracking and regulation of a wheeled mobile robot with controller/update law modularity. *IEEE Transactions on control systems technology*, 12(1):138–147, 2004.

[30] Evan Drumwright, John Hsu, Nathan Koenig, and Dylan Shell. Extending open dynamics engine for robotics simulation. In *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*, pages 38–50. Springer, 2010.

[31] Gijs Dubbelman and Frans CA Groen. Bias reduction for stereo based motion estimation with applications to large scale visual odometry. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2222–2229. IEEE, 2009.

[32] Gijs Dubbelman, Peter Hansen, and Brett Browning. Bias compensation in visual odometry. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2828–2835. IEEE, 2012.

[33] Gregory Dudek and Michael Jenkin. Inertial sensors, GPS, and odometry. In *Springer Handbook of Robotics*, pages 477–490. Springer, 2008.

[34] Bogdan Dumitrascu, Adrian Filipescu, Cristian Vasilache, Eugenia Minca, and Adriana Filipescu. Discrete-time sliding-mode control of four driving/steering wheels mobile platform. In *19th Mediterranean Conference on Control & Automation (MED)*, pages 1076–1081. IEEE, 2011.

[35] Matthew Dunbabin, Kane Usher, and Peter Corke. Visual motion estimation for an autonomous underwater reef monitoring robot. In *Field and Service Robotics*, pages 31–42. Springer, 2006.

[36] Gilberto Echeverria, Nicolas Lassabe, Arnaud Degroote, and Séverin Lemaignan. Modular open robots simulation engine: Morse. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 46–51. IEEE, 2011.

[37] Zheng Fang and Sebastian Scherer. Experimental study of odometry estimation methods using RGB-D cameras. In *International Conference on Intelligent Robots and Systems (IROS 2014), IEEE/RSJ*, pages 680–687. IEEE, 2014.

[38] Sara Farboud-Sheshdeh, Timothy D Barfoot, and Raymond H Kwong. Towards estimating bias in stereo visual odometry. In *Canadian Conference on Computer and Robot Vision (CRV)*, pages 8–15. IEEE, 2014.

[39] Friedrich Fraundorfer and Davide Scaramuzza. Visual odometry: Part ii: Matching, robustness, optimization, and applications. *IEEE Robotics & Automation Magazine*, 19 (2):78–90, 2012.

[40] Willow Garage. Universal Robot Description Format (URDF). *Http://Www. ros. org/urdf/*, 2009.

[41] Willow Garage. Robot operating system (ros), 2012.

[42] Rufino García-García, Miguel A Sotelo, Ignacio Parra, Daniel Fernández, José Eugenio Naranjo, and Miguel Gavilán. 3d visual odometry for road vehicles. *Journal of Intelligent & Robotic Systems*, 51(1):113–134, 2008.

[43] Andreas Geiger, Julius Ziegler, and Christoph Stiller. Stereoscan: Dense 3d reconstruction in real-time. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 963–968. Ieee, 2011.

[44] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361. IEEE, 2012.

[45] Jose Guivant, Eduardo Nebot, and Stephan Baiker. Autonomous navigation and map building using laser range sensors in outdoor applications. *Journal of robotic systems*, 17(10):565–583, 2000.

[46] Jose Guivant, Eduardo Nebot, and Hugh Durrant-Whyte. Simultaneous localization and map building using natural features in outdoor environments. In *Intelligent Autonomous Systems*, volume 6, pages 581–586, 2000.

[47] LIU Hongtao, Ruyi Jiang, HU Weng, and WANG Shigang. Navigational drift analysis for visual odometry. *Computing & Informatics*, 33(3), 2014.

[48] Albert S Huang, Abraham Bachrach, Peter Henry, Michael Krainin, Daniel Maturana, Dieter Fox, and Nicholas Roy. Visual odometry and mapping for autonomous flight using an RGB-D camera. In *Robotics Research*, pages 235–252. Springer, 2017.

[49] Bengt Erland Ilon. Wheels for a course stable selfpropelling vehicle movable in any desired direction on the ground or some other base, April 8 1975. US Patent 3,876,255.

[50] Jared Jackson. Microsoft Robotics Studio: A Technical Introduction. *IEEE Robotics & Automation Magazine*, 14(4), 2007.

[51] J-SR Jang. Anfis: adaptive-network-based fuzzy inference system. *IEEE Transactions on systems, man, and cybernetics*, 23(3):665–685, 1993.

[52] Ruyi Jiang, Reinhard Klette, and Shigang Wang. Statistical modeling of long-range drift in visual odometry. In *Asian Conference on Computer Vision*, pages 214–224. Springer, 2010.

[53] Wolfgang Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 32(5):922–923, 1976.

[54] Bernd Kitt, Andreas Geiger, and Henning Lategahn. Visual odometry based on stereo image sequences with RANSAC-based outlier rejection scheme. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 486–492. IEEE, 2010.

[55] Min Hyuc Ko, Beom-Sahng Ryuh, Kyoung Chul Kim, Abhijit Suprem, and Nitaigour P Mahalik. Autonomous greenhouse mobile robot driving strategies from system integration perspective: Review and application. *IEEE/ASME Transactions on Mechatronics*, 20(4):1705–1716, 2015.

[56] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004)*, volume 3, pages 2149–2154. IEEE, 2004.

[57] Mickaël Laîné, Silvia Cruciani, Emanuele Palazzolo, Nathan J Britton, Xavier Cavarelli, and Kazuya Yoshida. Navigation system for a small size lunar exploration rover with a monocular omnidirectional camera. In *First International Workshop on Pattern Recognition*, pages 100111M–100111M. International Society for Optics and Photonics, 2016.

[58] Ho Jae Lee, Jin Bae Park, and Guanrong Chen. Robust fuzzy control of nonlinear systems with parametric uncertainties. *IEEE Transactions on fuzzy systems*, 9(2):369–379, 2001.

[59] Thomas Linner, Alaguraj Shrikathiresan, Maxim Vetrenko, Bernhard Ellmann, and Thomas Bock. Modeling and operating robotic environent using Gazebo/ROS. In *Proceedings of the 28th international symposium on automation and robotics in construction (ISARC2011)*, pages 957–962.

[60] Kok-Lim Low. Linear least-squares optimization for point-to-plane ICP surface registration. *Chapel Hill, University of North Carolina*, 4, 2004.

[61] Zhiguo Lu, Chong Liu, Jun Peng, Habin Zhao, and Hong Wang. Motion simulation platform for a humanoid robot base on MATLAB. In *11th World Congress onIntelligent Control and Automation (WCICA)*, pages 13–18. IEEE, 2014.

[62] Edgar A Martinez-Garcia, Erik Lerin-Garcia, and Rafael Torres-Cordoba. A multi-configuration kinematic model for active drive/steer four-wheel robot structures. *Robotica*, 34(10):2309–2329, 2016.

[63] Antonio Matta-Gómez, Jaime Del Cerro, and Antonio Barrientos. Multi-robot data mapping simulation by using Microsoft Robotics Developer Studio. *Simulation Modelling Practice and Theory*, 49:305–319, 2014.

[64] Larry Matthies and STEVENA Shafer. Error modeling in stereo navigation. *IEEE Journal on Robotics and Automation*, 3(3):239–248, 1987.

[65] Syed Atif Mehdi and Karsten Berns. Behavior-based search of human by an autonomous indoor mobile robot in simulation. *Universal access in the information society*, 13 (1):45–58, 2014.

[66] SS Mehta, TF Burks, and WE Dixon. Vision-based localization of a wheeled mobile robot for greenhouse applications: A daisy-chaining approach. *Computers and electronics in agriculture*, 63(1):28–37, 2008.

[67] Johannes Meyer, Alexander Sendobry, Stefan Kohlbrecher, Uwe Klingauf, and Oskar Von Stryk. Comprehensive simulation of quadrotor UAVs using ROS and Gazebo. In *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*, pages 400–411. Springer, 2012.

[68] David S Michal and Letha Etzkorn. A comparison of Player/Stage/Gazebo and Microsoft Robotics Developer Studio. In *Proceedings of the 49th Annual Southeast Regional Conference*, pages 60–66. ACM, 2011.

[69] Olivier Michel. Webots: Symbiosis between virtual and real mobile robots. In *International Conference on Virtual Worlds*, pages 254–263. Springer, 1998.

[70] Adept MobileRobots. Mobile robots, seekur outdoor platform for robot research.

[71] Kevin L Moore and Nicholas S Flann. A six-wheeled omnidirectional autonomous mobile robot. *IEEE Control Systems*, 20(6):53–66, 2000.

[72] Thomas Moore and Daniel Stouch. A generalized extended kalman filter implementation for the robot operating system. In *Intelligent Autonomous Systems 13*, pages 335–348. Springer, 2016.

[73] L NatureWorks. Ingeo biopolymer 4043d technical data sheet, 2013.

[74] AJ Neal. Tips for selecting dc motors for your mobile robot, 2012.

[75] David Nistér, Oleg Naroditsky, and James Bergen. Visual odometry. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2004)*, volume 1, pages I–I. Ieee, 2004.

[76] Noboru Noguchi, Jeff Will, John Reid, and Qin Zhang. Development of a master–slave robot system for farm operations. *Computers and Electronics in agriculture*, 44(1): 1–19, 2004.

[77] Reza Oftadeh, Reza Ghabcheloo, and Jouni Mattila. A novel time optimal path following controller with bounded velocities for mobile robots with independently steerable wheels. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4845–4851. IEEE, 2013.

[78] T Oksanen and A Visala. Optimal control of tractor-trailer system in headlands. In *Proceedings of the 2004 Conference on Automation Technology for Off-Road Equipment*, page 255. American Society of Agricultural and Biological Engineers, 2004.

[79] François Pomerleau, Francis Colas, Roland Siegwart, and Stéphane Magnenat. Comparing ICP variants on real-world data sets. *Autonomous Robots*, 34(3):133–148, 2013.

[80] François Pomerleau, Francis Colas, Roland Siegwart, et al. A review of point cloud registration algorithms for mobile robotics. *Foundations and Trends® in Robotics*, 4 (1):1–104, 2015.

[81] Wei Qian, Zeyang Xia, Jing Xiong, Yangzhou Gan, Yangchao Guo, Shaokui Weng, Hao Deng, Ying Hu, and Jianwei Zhang. Manipulation task simulation using ROS and Gazebo. In *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 2594–2598. IEEE, 2014.

[82] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source Robot Operating System. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, 2009.

[83] Joern Rehder, Kamal Gupta, Stephen Nuske, and Sanjiv Singh. Global pose estimation with limited gps and long range visual odometry. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 627–633. IEEE, 2012.

[84] Shabnam Rehman, Syed Johar Raza, Andrew P Stegemann, Kevin Zeeck, Rakeeba Din, Amanda Llewellyn, Lynn Dio, Michael Trznadel, Yong Won Seo, Ashirwad J Chowriappa, et al. Simulation-based robot-assisted surgical training: a health economic evaluation. *International Journal of Surgery*, 11(9):841–846, 2013.

[85] RN RN Jorgensen, CG Sorensen, J Maagaard, Ib Havn, Kjeld Jensen, HT Sogaard, and LB Sorensen. Hortibot: A system design of a robotic tool carrier for high-tech plant nursing. 2007.

[86] Eric Rohmer, Surya PN Singh, and Marc Freese. V-REP: A versatile and scalable robot simulation framework. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1321–1326. IEEE, 2013.

[87] Sven Rönnbäck. *Developement of a INS/GPS navigation loop for an UAV*. 2000.

[88] Arno Ruckelshausen, Peter Biber, Michael Dorna, Holger Gremmes, Ralph Klose, Andreas Linz, Florian Rahe, Rainer Resch, Marius Thiel, Dieter Trautz, et al. BoniRob–an

autonomous field robot platform for individual plant phenotyping. *Precision agriculture*, 9(841):1, 2009.

[89] Atsushi Sakai, Masahito Mitsuhashi, and Yoji Kuroda. Noise model creation for visual odometry with neural-fuzzy model. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5190–5195. IEEE, 2010.

[90] M Sarkar, S Nandy, SRK Vadali, Spandan Roy, and SN Shome. Modelling and simulation of a robust energy efficient AUV controller. *Mathematics and Computers in Simulation*, 121:34–47, 2016.

[91] Davide Scaramuzza and Roland Siegwart. Appearance-guided monocular omnidirectional visual odometry for outdoor ground vehicles. *IEEE Transactions on robotics*, 24 (5):1015–1026, 2008.

[92] Alistair J Scarfe, Rory C Flemmer, HH Bakker, and Claire L Flemmer. Development of an autonomous kiwifruit picking robot. In *4th International Conference on Autonomous Robots and Agents (ICARA 2009)*, pages 380–384. IEEE, 2009.

[93] Konstantin Schauwecker and Andreas Zell. On-board dual-stereo-vision for the navigation of an autonomous MAV. *Journal of Intelligent & Robotic Systems*, 74(1-2):1–16, 2014.

[94] Ulrich Schwesinger, Cedric Pradalier, and Roland Siegwart. A novel approach for steering wheel synchronization with velocity/acceleration limits and mechanical constraints. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5360–5366. IEEE, 2012.

[95] Nagham Shalal, Tobias Low, Cheryl McCarthy, and Nigel Hancock. Orchard mapping and mobile robot localisation using on-board camera and laser scanner data fusion–part a: Tree detection. *Computers and Electronics in Agriculture*, 119:254–266, 2015.

[96] Roland Siegwart, Illah Reza Nourbakhsh, and Davide Scaramuzza. *Introduction to autonomous mobile robots*. MIT press, 2011.

[97] Hugo Silva, A Bernardino, and Eduardo Silva. Probabilistic egomotion for stereo visual odometry. *Journal of Intelligent & Robotic Systems*, 77(2):265, 2015.

[98] Niko Sünderhauf, Kurt Konolige, Simon Lacroix, and Peter Protzel. Visual odometry using sparse bundle adjustment on an autonomous outdoor vehicle. In *Autonome Mobile Systeme 2005*, pages 157–163. Springer, 2006.

[99] Kenjiro Tadakuma, Riichiro Tadakuma, Keiji Nagatani, Kazuya Yoshida, Steve Peters, Martin Udengaard, and Karl Iagnemma. Crawler vehicle with circular cross-section unit to realize sideways motion. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2008)*, pages 2422–2428. IEEE, 2008.

[100] Richard Thrapp, Christian Westbrook, and Devika Subramanian. Robust localization algorithms for an autonomous campus tour guide. In *Proceedings 2001 ICRA IEEE International Conference on Robotics and Automation*, volume 2, pages 2065–2071. IEEE, 2001.

[101] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005.

[102] Xinghua Tian, Feng Gao, Chenkun Qi, Xianbao Chen, and Dan Zhang. External disturbance identification of a quadruped robot with parallel-serial leg structure. *International Journal of Mechanics and Materials in Design*, 12(1):109–120, 2016.

[103] Miguel Torres-Torriti, T Arredondo, and P Castillo-Pizarro. Survey and comparative study of free simulation software for mobile robots. *Robotica*, 34(04):791–822, 2016.

[104] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle

adjustment,a modern synthesis. In *International workshop on vision algorithms*, pages 298–372. Springer, 1999.

[105] Spyros G Tzafestas. *Introduction to mobile robot control*. Elsevier, 2013.

[106] Jan Wendel, Oliver Meister, Christian Schlaile, and Gert F Trommer. An integrated GPS/MEMS-IMU navigation system for an autonomous helicopter. *Aerospace Science and Technology*, 10(6):527–533, 2006.

[107] Serhan Yamacli and Huseyin Canbolat. Simulation of a SCARA robot with PD and learning controllers. *Simulation Modelling Practice and Theory*, 16(9):1477–1487, 2008.

[108] Liangliang Yang, Noboru Noguchi, and Ryosuke Takai. Development and application of a wheel-type robot tractor. *Engineering in Agriculture, Environment and Food*, 9(2): 131–140, 2016.

[109] Teddy Yap, Mingyang Li, Anastasios I Mourikis, and Christian R Shelton. A particle filter for monocular vision-aided odometry. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5663–5669. IEEE, 2011.

[110] Bok-Joong Yoon, Myung-Wook Park, and Jung-Ha Kim. Ugv (Unmanned Ground Vehicle) navigation method using GPS and compass. In *International Joint Conference (SICE-ICASE)*, pages 3621–3625. IEEE, 2006.

[111] Yawei Zhang, Yunxiang Ye, Zhaodong Wang, Matthew E Taylor, Geoffrey A Hollinger, and Qin Zhang. Intelligent in-orchard bin-managing system for tree fruit production. In *Proceedings of the 2015 IEEE International Conference on Robotics and Automation, Piscataway, NJ, USA*, pages 26–30, 2015.